

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:)
Yosuke SENTA, et al.)
Serial No.: To be assigned) Group Art Unit: Unassigned
Filed: April 16, 2001) Examiner: Unassigned
For: CONTROL PROGRAM DEVELOPMENT)
SUPPORT APPARATUS)



**SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN
APPLICATION IN ACCORDANCE
WITH THE REQUIREMENTS OF 37 C.F.R. §1.55**

*Assistant Commissioner for Patents
Washington, D.C. 20231*

Sir:

In accordance with the provisions of 37 C.F.R. §1.55, the applicant submits herewith a certified copy of the following foreign application:

Japanese Patent Application No. 2000-249521
Filed: August 21, 2000.

It is respectfully requested that the applicant be given the benefit of the foreign filing date as evidenced by the certified papers attached hereto, in accordance with the requirements of 35 U.S.C. §119.

Respectfully submitted,
STAAS & HALSEY LLP

Date: April 16, 2001

By: _____

James D. Halsey, Jr.
Registration No. 22,729

700 11th Street, N.W., Ste. 500
Washington, D.C. 20001
(202) 434-1500

日 本 国 特 許 庁

PATENT OFFICE
JAPANESE GOVERNMENT



別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日

Date of Application:

2000年 8月21日

出 願 番 号

Application Number:

特願2000-249521

出 願 人

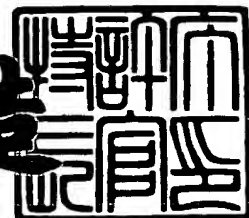
Applicant(s):

富士通株式会社

2001年 1月19日

特許庁長官
Commissioner,
Patent Office

及 川 耕 造



出証番号 出証特2000-3113997

【書類名】 特許願

【整理番号】 0051086

【提出日】 平成12年 8月21日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 17/50

【発明の名称】 制御プログラム開発支援装置

【請求項の数】 5

【発明者】

【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

【氏名】 千田 陽介

【発明者】

【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

【氏名】 佐藤 裕一

【特許出願人】

【識別番号】 000005223

【氏名又は名称】 富士通株式会社

【代理人】

【識別番号】 100092978

【弁理士】

【氏名又は名称】 真田 有

【電話番号】 0422-21-4222

【手数料の表示】

【予納台帳番号】 007696

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9704824

【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 制御プログラム開発支援装置

【特許請求の範囲】

【請求項 1】 機構の動作を制御する制御プログラムを実行し、該機構に対する制御量を所定の制御周期で算出して出力する制御プログラム実行部と、

該機構を仮想的なモデルとして内部に構築され、該モデルを用い、前記所定制御周期よりも短く設定された所定のシミュレーション周期で、前記所定の制御周期に対応する時間に亘って該機構の動作をシミュレートすることにより、該機構の状態量を算出して出力するシミュレーション部と、

該制御プログラム実行部から該シミュレーション部への前記制御量、および、該シミュレーション部から該制御プログラム実行部への前記状態量を一時的に保持し中継する中継部と、

該シミュレーション部からの前記状態量が該中継部に保持されると、該シミュレーション部を、該制御プログラム実行部からの応答待ち状態へ移行させるとともに、該制御プログラム実行部による、前記状態量に応じた制御量の算出動作を開始させる一方、該制御プログラム実行部からの前記制御量が該中継部に保持されると、該制御プログラム実行部を、該シミュレーション部からの応答待ち状態へ移行させるとともに、該シミュレーション部による、前記制御量に応じたシミュレーション動作を開始させるシミュレーション制御部とをそなえたことを特徴とする、制御プログラム開発支援装置。

【請求項 2】 サーボ機構の動作を制御する制御プログラムを実行し、該サーボ機構に対する制御量を所定の制御周期で算出して出力する制御プログラム実行部と、

該サーボ機構を仮想的なモデルとして内部に構築され、該モデルを用いて該サーボ機構の動作を動力学的に解析しながら、前記所定制御周期よりも短く設定された所定のシミュレーション周期で、前記所定の制御周期に対応する時間に亘って該サーボ機構の動作をシミュレートすることにより、該サーボ機構の状態量を算出して出力するシミュレーション部と、

該制御プログラム実行部から該シミュレーション部への前記制御量、および、

該シミュレーション部から該制御プログラム実行部への前記状態量を一時的に保持し中継する中継部と、

該シミュレーション部からの前記状態量が該中継部に保持されると、該シミュレーション部を、該制御プログラム実行部からの応答待ち状態へ移行させるとともに、該制御プログラム実行部による、前記状態量に応じた制御量の算出動作を開始させる一方、該制御プログラム実行部からの前記制御量が該中継部に保持されると、該制御プログラム実行部を、該シミュレーション部からの応答待ち状態へ移行させるとともに、該シミュレーション部による、前記制御量に応じたシミュレーション動作を開始させるシミュレーション制御部とをそなえたことを特徴とする、制御プログラム開発支援装置。

【請求項 3】 該制御プログラム実行部が、一制御周期中において異なるタイミングで該シミュレーション部に入力されるべき複数の制御量を出力するものであり、

前記複数の制御量をそれぞれ所定のタイミングで該シミュレーション部に入力するように制御量の入力制御を行なうマルチレート制御手段をそなえたことを特徴とする、請求項 2 記載の制御プログラム開発支援装置。

【請求項 4】 該シミュレーション制御部が、該シミュレーション部によるシミュレーション結果に基づいて、該制御プログラム実行部による前記制御量の算出動作の開始タイミングを決定することを特徴とする、請求項 2 記載の制御プログラム開発支援装置。

【請求項 5】 該モデルが、その動作のシミュレーションを個別に実行することが可能な複数の部分から構成されるものであり、

該シミュレーション部が、前記複数の部分それぞれの動作を並列的にシミュレートする複数のプロセッサをそなえて構成されていることを特徴とする、請求項 2 記載の制御プログラム開発支援装置。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、比較的小型で応答の速い製品について制御プログラムのサーボ制御

部分（サーボ制御プログラム）の開発を支援する技術に関するもので、例えば、磁気ディスクドライブ（HDD）、光ディスクドライブ（CD、MO、DVD、MD）、磁気テープ装置（DAT、VTR）、NC工作機など、緻密なサーボ制御を必要とするあらゆる製品についてのサーボ制御プログラム（ファームウェア）を開発する際に用いて好適の、制御プログラム開発支援装置に関する。

【0002】

【従来の技術】

一般に、アクチュエータ（モータ）やセンサを有し3次元的な動作を行なう機構（メカ）を設計する際には、その機構の構想を練った後、詳細設計、出図、部品手配を行なってから、部品の組立を行なって実機を試作し、実機の動作等についての評価を行なう。そして、評価の結果に応じて設計変更を行なってから設計変更後の実機を試作し、再び評価を行なうという処理を繰り返し、評価の結果が良好であれば、設計を完了する。

【0003】

また、一般に、上述のごとく設計された機構を動作させるべく、その機構を制御するための制御プログラムを開発し、その制御プログラムを、制御対象の機構内に組み込まれるマイクロコンピュータ（以下、マイコンという場合がある）で実行させるようにしている。このマイコンが実行する上記制御プログラムのことを、以下、組込みソフトウェアという場合がある。

【0004】

上述のような制御プログラム（組込みソフトウェア）を開発する際、従来、制御すべき機構の試作品（実機）が完成している必要がある。即ち、試作が完了して初めてメカを具体的に動かすことができ、それを使って組込みソフトウェアの開発を開始することができるわけである。

【0005】

この組込みソフトウェアの開発は、試作品の完成後、その試作品を実際に動作させながら、以下の手順で行なわれる。つまり、まず、組込みソフトウェアの概略設計を行ない、その概略設計に基づいて詳細設計を行なってから、詳細設計の結果をコーディングして組込みソフトウェアを作成し、その組込みソフトウェア

のデバッグを行なう。

【0006】

【発明が解決しようとする課題】

ところで、ファームウェアのサーボ機構部分の構築・検証も、従来、上述した組込みソフトウェア開発の場合と同様、実際のメカ（実機）を用いて行なわれている。

しかし、ファームウェアのサーボ機構部分の構築・検証に上述の手法を用いた場合、実機が完成しないと検証を行なうことができず開発に時間やコストがかかり非効率的であるという課題があるほか、実機を用いて検証を行なうため、イベントブレイク等でファームウェアプログラムを停止しても、サーボ機構を成すモータは回転し続け、ステップデバッグを行なうことができないなどの課題もあった。

【0007】

また、月産数万台も生産される製品に対するサーボ機構制御プログラムは、ある程度のメカのバラツキに対応できなくてはならない。実際のメカを用いた開発では、上記バラツキを考慮した所望の状態のメカを入手することは困難であり、ファームウェアがどの程度のバラツキまで対応できているかを知ることは困難であった。

【0008】

そこで、実際のメカの代わりに計算機上に仮想的なモデルを作成し、このモデルを実際のファームウェアに制御させることにより、実機を用いることなく効率的にファームウェアの開発（構築・検証）を行なえるようにすることが望まれている。このような技術が実現されると、ファームウェアの先行開発が可能になるほか、実際のモータを用いないためステップデバッグなどの機能を使用することが可能であり、新しいアクチュエータやセンサを用いた新規の制御手法を、実機を作成することなく簡単に検証可能になるなどの利点を得られる。

【0009】

従来、例えばdSPACE社のHIL（Hardware In the Loop）シミュレーションシステムのように、実際の製品の代わりとなるシミュレータ内のモデルに対

し、別のコンピュータで動作する制御プログラムによってそのモデルの制御を行なうものは存在している。しかし、このシステムは、実時間でシミュレーションを行なうものであるため、自動車や船など応答の遅い製品（例えば 1 msec 以上のシミュレーション時間間隔でシミュレーション可能な製品）がシミュレーション対象となっている。従って、上述のシステムにより、磁気ディスクドライブ（HDD）など比較的小型で応答の速い製品（例えば 100 μ sec 程度のシミュレーション時間間隔でシミュレーションを行なうべき製品）のシミュレーションを行なうのは困難である。

【0010】

また、本願発明者等は、3次元リアルタイムシミュレーション装置を中核に据え、メカ試作品を作らなくても組み込みソフトウェア（制御プログラム）の開発をメカ設計とは単独に進めることができるような支援システムを提案している。この支援システムによる制御プログラムの検証はタスクレベルで行なわれている。つまり、この支援システムでは、制御プログラム側からの指令に応じたアクチュエータ動作がシミュレートされ、その動作に従ったセンサのオン／オフ信号に基づいて、制御プログラムの検証が行なわれている。従って、この支援システムでは、モデルの動力学を解析するサーボレベルで制御プログラムの検証（開発・デバッグ）を行なうことができない。

【0011】

サーボ機構は、機械的運動のための自動フィードバック制御システムであり、制御量または制御出力が機械的な位置〔または誘導変数（速度加速度など）の一つ〕であるようなシステムを制御するために用いられるものである。このため、サーボ制御部分の制御プログラムの検証を行なうためには、サーボ機構のシミュレーションを、タスクレベルではなく、前記機械的な位置等の制御量を動力的に解析しながら（つまりサーボレベルで）行なう必要がある。サーボレベルで検証を行なう場合、動力学解析を行なって厳密なシミュレーションを行なう必要があるため、実時間でシミュレーションを行なうことは困難になることがある。

【0012】

従って、実際のメカを用いることなく、比較的小型で応答の速い製品について

制御プログラムのサーボ制御部分（以下、サーボ制御プログラムという場合がある）の開発・デバッグ（検証）を行なえるようにすることが望まれている。

なお、従来、MATLABなどに代表される数値解析ソフトウェアを用いることにより、制御対象とその制御対象についての制御則とをモデル化し、その制御則の検証を行なう一般的な手法は存在している。この手法は、理論レベルで制御則の検証を行なうことはできるが、検証した制御則を実際のファームウェア（制御プログラム）にコード化する際に生じる様々な問題点（実行速度、コードサイズ、消費メモリ量、バグの混在など）には対応することができない。また、この手法は、制御則の検証を行なうものであって、その制御則に基づいて作成された制御プログラムの検証を行なうことはできず、その制御プログラムの検証に際しては、結局、前述した従来手法を用いることになる。

【0013】

本発明は、このような課題に鑑み創案されたもので、実際のメカを用いることなく、比較的小型で応答の速い製品（機構）を制御するための制御プログラムの開発・デバッグ（検証）を行なえるようにした、制御プログラム開発支援装置を提供することを目的とする。

【0014】

【課題を解決するための手段】

上記目的を達成するために、本発明の制御プログラム開発支援装置（請求項1，2）は、シミュレーション部での所定のシミュレーション周期が制御プログラム実行部での所定の制御周期よりも短く設定され、シミュレーション部が、前記所定の制御周期に対応する時間に亘って前記所定のシミュレーション周期で機構（サーボ機構）の動作をシミュレートし、そのシミュレーションによって得られた該機構の状態量を中継部へ出力するように構成されるとともに、シミュレーション制御部が、該シミュレーション部からの前記状態量が該中継部に保持されると、該シミュレーション部を、該制御プログラム実行部からの応答待ち状態へ移行させるとともに、該制御プログラム実行部による、前記状態量に応じた制御量の算出動作を開始させる一方、該制御プログラム実行部からの前記制御量が該中継部に保持されると、該制御プログラム実行部を、該シミュレーション部からの

応答待ち状態へ移行させるとともに、該シミュレーション部による、前記制御量に応じたシミュレーション動作を開始させることを特徴としている。

【0015】

上述した本発明の制御プログラム開発支援装置では、制御対象プラントである機構（サーボ機構）がモデル化され、そのモデルの動作をシミュレートするためのシミュレーション部と、その機構（サーボ機構）の動作を制御するファームウェアを実行する制御プログラム実行部とが、中継部を介して接続される。そして、シミュレーション部の動作と制御プログラム実行部の動作とを、シミュレーション制御部によって同期させることができるので、時間厳密性を保ったまま精密なシミュレーションを行なうことが可能になる。その際、シミュレーション部が、制御プログラム実行部での制御周期（サーボ制御ファームウェアのサーボ周期）よりも短いシミュレーション周期でシミュレーションを行なうので、精密なシミュレーションを実行することが可能になる。

【0016】

なお、該シミュレーション制御部（同期処理部）の同期設定を行なうための同期設定手段をさらにそなえ、その同期設定手段を、グラフィカルユーザインタフェース機能を用いて構成してもよい。これにより、シミュレーション部の動作と制御プログラム実行部の動作との同期設定を行なうことができ、さらに、グラフィックを用いて容易にその同期設定を行なうことができる。

【0017】

また、該制御プログラム実行部が、一制御周期中において異なるタイミングで該シミュレーション部に入力されるべき複数の制御量を出力するように構成されるとともに、前記複数の制御量をそれぞれ所定のタイミングで該シミュレーション部に入力するように制御量の入力制御を行なうマルチレート制御手段がそなえられていてもよい（請求項2）。このとき、該マルチレート制御手段の設定を行なうためのマルチレート設定手段をそなえ、そのマルチレート設定手段を、グラフィカルユーザインタフェース機能を用いて構成してもよい。

【0018】

上述したマルチレート制御手段を用いることにより、シミュレーション部は、

一制御周期中に制御量が変化するマルチレート制御をシミュレートすることができる。また、マルチレート設定手段により、そのマルチレート制御を定義・設定することができ、さらに、グラフィックを用いて容易にそのマルチレート制御の設定を行なうことができる。

【0019】

さらに、該シミュレーション制御部が、該シミュレーション部によるシミュレーション結果に基づいて、該制御プログラム実行部による前記制御量の算出動作の開始タイミングを決定してもよい（請求項3）。これにより、シミュレーション部によるシミュレーション結果に応じて、制御プログラム実行部でのサーボ制御ルーチンへ移行することができ、タイマによるフェイルセーフ機能を確認したり、単位時間当たりの変化量（速度、回転数等）を測定する処理などに対応したりすることができる。

【0020】

また、該モデルが、その動作のシミュレーションを個別に実行することが可能な複数の部分から構成されるものであり、該シミュレーション部が、前記複数の部分それぞれの動作を並列的にシミュレートする複数のプロセッサをそなえて構成されていてもよい（請求項4）。これにより、モデルの構成部分の動作シミュレーションを、複数のプロセッサで並列的に実行することができる。

【0021】

さらに、該中継部を、該制御プログラム実行部から該シミュレーション部への前記制御量と該シミュレーション部から該制御プログラム実行部への前記状態量とを含むデータを一時的に保持しうる複数のレジスタと、該複数のレジスタと該制御プログラム実行部との間で前記データの書込／読出を制御する第1書込／読出制御部と、該複数のレジスタと該シミュレーション部との間で前記データの書込／読出を制御する第2書込／読出制御部とによって構成してもよい（請求項5）。

【0022】

これにより、制御プログラム実行部からの制御量は、第1書込／読出制御部により適当なレジスタに一旦書き込まれた後、同期信号に応じ、第2書込／読出制

御部により読み出されてシミュレーション部に入力される。一方、シミュレーション部からの状態量は、第 2 書込／読出制御部により適当なレジスタに一旦書き込まれた後、同期信号（割込信号）に応じ、第 1 書込／読出制御部により読み出されて制御プログラム実行部に入力される。

【 0 0 2 3 】

このとき、該制御プログラム実行部による前記制御量の算出動作を開始させるべく該シミュレーション部から該複数のレジスタの一つに入力された割込信号については、該第 1 書込／読出制御部を介することなく、当該レジスタから該制御プログラム実行部へ直接的に送出してもよい。これにより、制御プログラム実行部側では、第 1 書込／読出制御部による読出制御（ファームウェアの読み込み動作）を行なうことなく割込信号（同期信号）を得ることができ、ハードウェア割り込みを利用した同期処理を行なうことが可能になる。

【 0 0 2 4 】

また、該複数のレジスタに保持されているデータを表示しうるデータ表示部をそなえてもよい。このとき、該複数のレジスタの中から選択した少なくとも一つのレジスタに保持されているデータを該データ表示部に表示させる選択部をそなえてもよいし、該データ表示部を該複数のレジスタのうちの特定のものに直接的に接続し、このデータ表示部において、該特定のレジスタに保持されているデータを表示してもよい。これにより、制御プログラム実行部とシミュレーション部との間で通信中のデータがデータ表示部で表示され、オペレータ等はそのデータを参照・確認することができる。

【 0 0 2 5 】

さらに、該複数のレジスタのうちの少なくとも一つに所望のデータを強制的に設定・格納するためのデータ入力部をそなえてもよい。このとき、該データ入力部を、該複数のレジスタのうちの特定のものに直接的に接続し、このデータ入力部から、該特定のレジスタに前記所望のデータを設定してもよい。これにより、オペレータ等は、任意のデータを、データ入力部からレジスタに書き込むことによって制御プログラム実行部やシミュレーション部へ直接的に入力することができ、そのデータに応じた制御プログラムの挙動やモデルの動作状態を確認するこ

とができる。

【0026】

またさらに、該複数のレジスタのうちの少なくとも一つから読み出されたデータにノイズを重畳するノイズ重畳部をそなえてもよい。これにより、オペレータ等は、ノイズをノイズ重畳部によってデータに重畳することができ、そのノイズに応じた制御プログラムの挙動やモデルの動作状態を確認することができる。

【0027】

【発明の実施の形態】

以下、図面を参照して本発明の実施の形態を説明する。

〔1〕本発明の一実施形態の説明

〔1-1〕本実施形態の全体構成

図1は本発明の一実施形態としての制御プログラム開発支援装置の構成を従来システムの構成と比較して示すもので、図1(a)は従来システムの構成を示すブロック図、図1(b)は本実施形態の構成を示すブロック図である。

【0028】

図1(a)に示すように、サーボ制御プログラムの開発・デバッグ(検証)を行なうための従来システムでは、実際のサーボ機構、つまり、実際のメカ100および制御回路200を用いている。

メカ100は、アクチュエータ110およびセンサ120を含んで構成されている。アクチュエータ110は、制御回路200からの制御量に従ってメカ100の動作を制御するものであり、センサ120は、アクチュエータ110で制御されるとともに外乱を受けるメカ100の動作状態を状態量として検出するものである。

【0029】

制御回路200は、制御用LSI210、ドライバ220および検出回路230をそなえて構成されている。

制御用LSI210は、サーボ機構の動作を制御する制御プログラム(制御プログラムのサーボ制御部分/ファームウェア)を実行するもので、MCU(MicroController Unit)211、メモリ212、サーボロジック213およびセンサ

ロジック 2 1 4 を有している。

【 0 0 3 0 】

メモリ 2 1 2 は、前記制御プログラムを含む各種情報を格納するものであり、MCU 2 1 1 は、いわゆるワンチップマイコンであり、メモリ 2 1 2 に格納された制御プログラムを実行し、メカ 1 0 0（サーボ機構）に対する制御量を、メカ 1 0 0 側からの状態量（センサ 1 2 0 による検出結果）に応じて演算するものである。

【 0 0 3 1 】

サーボロジック 2 1 3 およびセンサロジック 2 1 4 は、それぞれ、ドライバ 2 2 0 および検出回路 2 3 0 に接続されるもので、例えば A/D（アナログ／デジタル変換器）、D/A（デジタル／アナログ変換器）、P I O（Parallel Input /Output）などの一般的なロジック回路を含んで構成されるものである。また、サーボロジック 2 1 3 としては、パルス幅変調（PWM）信号発生器を用いることもできる。

【 0 0 3 2 】

ドライバ 2 2 0 は、サーボロジック 2 1 3 からの制御信号（制御量）に基づいてアクチュエータを駆動するものであり、検出回路 2 3 0 は、メカ 1 0 0 のセンサ 1 2 0 により検出された状態量を受け取り、センサロジック 2 1 4 に入力するものである。

なお、メカ 1 0 0 や制御回路 2 0 0 の状態は、オシロスコープ等の状態表示部 3 0 0 によって表示されるようになっている。

【 0 0 3 3 】

上述した従来システムでは、実際のメカ 1 0 0 や制御回路 2 0 0 が用いられ、MCU 2 1 1 が、検出回路 2 3 0 およびセンサロジック 2 1 4 を介して得た情報（センサ 1 2 0 により検出された状態量）に基づいてサーボ演算を行ない、その演算結果である制御量を、制御信号としてサーボロジック 2 1 3 およびドライバ 2 2 0 を介しアクチュエータ 1 1 0 に指令を与えることで、サーボループが構築されている。

【 0 0 3 4 】

これに対し、図 1 (b) に示すごとく、本実施形態の制御プログラム開発支援装置 1 は、制御回路 1 0 とモデル実行環境 2 0 とを、中継回路 3 0 を介して通信可能に接続することによって構成されており、制御回路 1 0 およびモデル実行環境 2 0 は、それぞれ、実際にはファームウェア実行用プロセッサおよびモデル演算用プロセッサにより構成されている。

【0035】

そして、制御回路 1 0 は、サーボ機構の動作を制御する制御プログラム（以下、制御ファームウェアという場合もある）を実行しそのサーボ機構に対する制御量を所定の制御周期（制御ルーチンの呼び出し間隔） ΔT で算出して出力する制御プログラム実行部として機能するもので、MCU 1 2 およびメモリ 1 3 を含む制御用 LSI 1 1 から構成されている。

【0036】

ここで、メモリ 1 3 は、前記制御プログラムを含む各種情報を格納するものであり、MCU 1 2 は、いわゆるワンチップマイコンであり、メモリ 1 3 に格納された制御プログラムを実行し、サーボ機構（本実施形態では後述する仮想メカモデル 2 1）に対する制御量を、モデル実行環境 2 0 側からの状態量（シミュレーション結果）に応じて演算するものである。

【0037】

モデル実行環境（以下、シミュレータという場合もある）2 0 は、サーボ機構を仮想的なモデル（仮想メカモデル）2 1 として内部に構築され、その仮想メカモデル 2 1 を用いてサーボ機構の動作を動力的に解析しながら所定のシミュレーション周期 Δt でサーボ機構の動作をシミュレートすることにより、サーボ機構の状態量を算出して出力するシミュレーション部として機能するものである。

【0038】

ここで、仮想メカモデル 2 1 は、図 1 (a) に示した従来システムにおけるサーボ機構の部分、即ち、サーボロジック 2 1 3、ドライバ 2 2 0、メカ 1 0 0（アクチュエータ 1 1 0 およびセンサ 1 2 0 を含む）、検出回路 2 3 0 およびセンサロジック 2 1 4 の部分をモデル化したものである。

【0039】

中継回路（中継部）30は、制御回路10からモデル実行環境20への制御量、および、モデル実行環境20から制御回路10への状態量を中継すべく、これらの制御量や状態量を一時的に格納・保持する共有メモリ（バッファ、レジスタ）31を有して構成されている。この中継回路30の詳細構成については、図16～図20を参照しながら後述する。

状態表示記録部（データ表示部）40は、モデル実行環境20によるシミュレーション結果を表示したり記録したりするものである。

【0040】

そして、本実施形態の制御プログラム開発支援装置1においては、シミュレーション周期 Δt が制御周期 ΔT よりも短く設定され、モデル実行環境20は、所定の制御周期 ΔT に対応する時間に亘って所定のシミュレーション周期 Δt でシミュレーションを行ない、そのシミュレーションによって得られたサーボ機構の状態量を中継回路30へ出力するように構成されている。

【0041】

さらに、本実施形態の制御プログラム開発支援装置1には、後述するようにして制御回路10の動作とモデル実行環境20の動作との同期処理を行なうシミュレーション制御部（同期処理部）22がそなえられている。このシミュレーション制御部22は、図1（b）ではモデル実行環境20にそなえられているが、実際には、制御回路10およびモデル実行環境20にまたがる形で配置され、図2、図3および図5～図10にて後述する手法に従い、ソフトウェアにより実現されるものである。

【0042】

このシミュレーション制御部22は、モデル実行環境20からの状態量が中継回路30の共有メモリ31に保持されると、モデル実行環境20を、制御回路10からの応答待ち状態へ移行させるとともに、制御回路10による、状態量に応じた制御量の算出動作を開始させる一方、制御回路10からの制御量が中継回路30に保持されると、制御回路10を、モデル実行環境20からの応答待ち状態へ移行させるとともに、モデル実行環境20による、制御量に応じたシミュレーション動作を開始させるように機能する。その際、図2や図3に示すごとく、制

御回路（制御ファームウェア）10とモデル実行環境（シミュレータ）20との間では、サーボ割込信号（SVInt：Servo Interrupt）やサーボタスク信号（Servo Task）がやり取りされる。

【0043】

図1（b）に示す本実施形態の制御プログラム開発支援装置1においては、MCU12、メモリ13など制御プログラムを動作させるために必要な環境（図1には示していないが浮動点演算装置などの補助回路も含む）以外の部分を全てモデル化し、モデル化された仮想メカモデル21の動作がモデル実行環境（シミュレータ）20によりシミュレートされ、仮想メカモデル21の状態量が算出される。そして、モデル実行環境20と制御回路10との間で、中継回路30の共有メモリ31を介して通信を行なうことにより、サーボループが構築されている。

【0044】

このとき、対象とするメカの応答の高速化に伴いモデル実行（シミュレーション）のサンプリング間隔 Δt を細かくする必要があるが生じたり、モデル21を精密化するのに伴い計算量が増大したりすると、シミュレータ20において、実時間でモデル演算を行なうことが難しくなる。

【0045】

そこで、本実施形態の制御プログラム開発支援装置1においては、MCU12の演算を遅らせ、同期信号（サーボ割込信号）SVIntを用いてMCU12の演算と仮想メカモデル21の演算とを同期させ、サーボ特性を変化させずにスローモーション的にシミュレーションを行なうようにしている。さらに、本実施形態では、サーボロジック213がマルチレート制御を実現していた場合の対応手法についても、図11～図13を参照しながら説明する。

【0046】

〔1-2〕基本的な同期処理手順

次に、図2および図3を参照しながら、本実施形態における前記シミュレーション制御部（同期処理部）22の動作（同期処理手順）について説明する。図2および図3は、それぞれ、本実施形態での同期処理手順（シミュレーション制御部22の動作）を説明するためのフローチャートおよびタイムチャートである。

【0047】

図2に示す制御ファームウェア（制御F/Wの部分；制御回路10で実行される制御プログラム）は、サーボ制御ルーチン部分であって、一般にタイマ割り込みを用いて一定間隔（ ΔT ）で呼び出されたり、あるいは、制御対象が例えば磁気ディスクドライブ（HDD）である場合にはサーボマークを磁気ディスクから読み込んだ時に発生する割込信号によって呼び出されたりする。

【0048】

本実施形態の制御プログラム開発支援装置1では、サーボ制御ルーチンを呼び出すための割り込み要因を、シミュレータ20側からの割込信号（SVInt）に変更し、シミュレータ20によるシミュレーション終了（所定間隔のモデル実行の完了）に伴って、サーボ制御ルーチンが呼び出されるようにしている。

【0049】

なお、割込信号を用いず、メインループにてシミュレータ20におけるSVIntの状態を常に監視するように構成してもよい。これにより、図3を参照して後述するごとく、制御ルーチンとシミュレーションとを交互に実行することができる。

【0050】

図2に示すように、制御ファームウェア（制御回路10）においてはシミュレータ20からの割込信号（SVInt）に応じてステップS11～S16の処理が実行され、シミュレータ20においては制御回路10からのサーボタスク信号に応じてステップS21～S27が実行される。これにより、本実施形態の同期処理（シミュレーション制御部22）が実現され、制御回路10の動作とシミュレータ20の動作との同期処理が行なわれる。

【0051】

具体的に説明すると、シミュレータ20において割込信号SVIntがセットされると（ステップS21）、制御回路10では、制御ファームウェアが起動され、サーボタスク信号がセットされる（ステップS11）。

このサーボタスク信号は、シミュレータ20に通知され、シミュレータ20では、サーボタスク信号のセット検知に伴い（ステップS22のYESルート）、

割込信号 $SVInt$ をクリアする（ステップ $S23$ ）。この後、シミュレータ 20 は、制御回路 10 からのサーボタスク信号がクリアされるまで、つまり、ステップ $S24$ で YES 判定となるまで、制御ルーチン終了（サーボタスク信号クリア）待ち状態となる。

【0052】

また、制御回路 10 では、割込信号 $SVInt$ のクリアを検知すると（ステップ $S12$ の YES ルート）、中継回路 30 の共有メモリ（レジスタ）31 に保持されている状態量（シミュレータ 20 によるシミュレーション結果）が読み込まれ（ステップ $S13$ ）、サーボ演算、つまり状態量に応じた制御量の演算処理が実行される（ステップ $S14$ ）。そして、算出された制御量を中継回路 30 へ出力して共有メモリ 31 に格納してから（ステップ $S15$ ）、サーボタスク信号をクリアし（ステップ $S16$ ）、サーボ制御ルーチンを終了する。

【0053】

一方、シミュレータ 20 では、サーボタスク信号のクリアを検知すると（ステップ $S24$ の YES ルート）、中継回路 30 の共有メモリ（レジスタ）31 に保持されている制御量が読み込まれ（ステップ $S25$ ）、シミュレーションが行なわれ（ステップ $S26$ ）、その制御量に応じた仮想メカモデル 21 の状態量の演算処理が実行される。そして、シミュレーション結果として得られた状態量を中継回路 30 へ出力して共有メモリ 31 に格納すると（ステップ $S27$ ）、再び、割込信号 $SVInt$ がセットされ（ステップ $S21$ ）、以降、上述と同様の処理が繰り返されることになる。

【0054】

図 2 にて上述した同期処理について図 3 を参照しながら説明する。図 3 に示すように、シミュレータ 20 は、時刻 $t1$ で状態量を出力してシミュレーションを終了すると（ステップ $S27$ ）、制御ルーチン終了（サーボタスク信号クリア）待ちの状態に移行するとともに、時刻 $t2$ で割込信号 $SVInt$ をセットし（ステップ $S21$ ）、制御ファームウェアを呼び出して起動する。

【0055】

そして、時刻 $t3$ において制御回路 10 がサーボタスク信号をセットすると（

ステップS11)、そのサーボタスク信号に応じ、時刻 t_4 においてシミュレータ20が割込信号SVIntをクリアし(ステップS23)、制御ファームウェア(HDD F/W)は、時刻 t_5 から制御ルーチン実行中となる(ステップS13~S15)。

【0056】

この後、制御ファームウェアは、時刻 t_6 で制御量を出力してサーボ演算を終了すると(ステップS15)、シミュレーション終了(SVIntセット)待ちの状態に移行するとともに、時刻 t_7 でサーボタスク信号をクリアする(ステップS16)。これに伴い、シミュレータ20は、時刻 t_8 からシミュレーション実行中となり(ステップS25~S27)、時刻 t_9 で状態量を出力してシミュレーションを終了すると(ステップS27)、前述した処理を繰り返すことになる。

【0057】

〔1-3〕制御周期とシミュレーション周期との関係

さて、制御ルーチンが本来呼び出される間隔である制御周期を ΔT とすると、従来、シミュレータ20でのシミュレーション(モデル実行)も、制御周期と同じ間隔 ΔT で行なわれていた。しかし、精度のよいシミュレーションを行なうためや、後述するマルチレート制御などに対応するためには、シミュレーション間隔(シミュレーション周期) Δt を、制御周期 ΔT よりも細かくする必要がある。

【0058】

そこで、本実施形態の制御プログラム開発支援装置1では、図2におけるシミュレーション実行部分(ステップS26)に際して、細かいシミュレーションサイクルを実行する毎に Δt を加算し、その合計時間が制御周期 ΔT に達したところで、シミュレーションループから抜けるようにしている。

【0059】

より具体的に説明すると、図4は本実施形態のシミュレータでのシミュレーション原理を説明するためのフローチャート(ステップS31~S34)であり、この図4に示すように、図2におけるシミュレーション実行部分(ステップS2

6) では、シミュレーションの開始に先立ち、合計時間 t が 0 に設定される (ステップ S 3 1)。この後、シミュレータ 2 0 では、シミュレーション間隔 Δt に対応する時間のシミュレーションが実行される都度 (ステップ S 3 2)、合計時間 t にシミュレーション周期 Δt が加算され (ステップ S 3 3)、その合計時間が制御周期 ΔT に到達したか否かが判定される (ステップ S 3 4)。合計時間 t が ΔT 未満である場合 (ステップ S 3 4 の NO ルート)、ステップ S 3 2 に戻る一方、合計時間 t が ΔT 以上となった場合 (ステップ S 3 4 の YES ルート)、シミュレーションを終了する。

【 0 0 6 0 】

〔 1 - 4 〕 基本的な同期設定

そして、本実施形態の制御プログラム開発支援装置 1 のシミュレータ 2 0 には、上述した周期 ΔT を設定するための機能〔シミュレーション制御部 (同期処理部) 2 2 の同期設定を行なうための同期設定手段〕がそなえられている。この機能は、シミュレータ 2 0 におけるグラフィカルユーザインタフェース機能 (GUI 機能) を用いて実現されている。つまり、オペレータ等がシミュレータ 2 0 におけるディスプレイ (図示省略) 上の表示を参照しながらキーボードやマウス等を操作することにより、周期 ΔT の設定 (同期設定) が行なわれるようになっている。本実施形態では、一般的なモデル記述方法〔例えば MATLAB/Simulink (MathWorks 社)〕を用いて、ディスプレイ上に図 5 に示すごとく簡易なモデル (ブロック線図) をグラフィカルに記述・作成することにより、同期設定が行なわれる。

【 0 0 6 1 】

ここで、図 5 は本実施形態で同期設定を行なうためのモデル記述レベル (ディスプレイでの表示状態) を示す図、図 6 は図 5 に示すモデル記述レベルで記述・設定された同期ブロック B 2 の動作を説明するためのフローチャートである。

本実施形態では、図 5 に示すモデル記述レベルで、一定周期 ΔT の方形波 (パルス) を発生する方形波発生ブロック B 1 と同期ブロック B 2 とを作成し、方形波発生ブロック B 1 からのパルスを同期ブロック B 2 に入力するようにブロック B 1 および B 2 を記述する。

【0062】

同期ブロック（同期処理部，シミュレーション制御部22）B2は、方形波発生ブロックB1からのパルスのエッジ（立ち上がり）を検出し、そのエッジをトリガにして、制御ファームウェアとの同期処理を行なう。

この同期ブロックB2の動作を図6に示すフローチャート（ステップS41～S47）に従って具体的に説明する。なお、図6において、ステップS41およびS42はエッジ検出（パルス立ち上がり検出）を行なう部分である。また、ステップS44～S47は、制御ファームウェアとの同期を行なう部分で、図2のステップS21～S24に対応する処理である。

【0063】

同期ブロックB2は、シミュレーション周期 Δt 毎に起動され、その都度、まず、同期ブロックB2に対する現入力（方形波発生ブロックB1からのパルス）が“High”であるか否かを判断する（ステップS41）。現入力“High”でなければ、即ち“Low”である場合（ステップS41のNOルート）、現入力“Low”を前入力に置き換えて（ステップS43）、処理を終了する。

【0064】

一方、現入力“High”である場合（ステップS41のYESルート）、前入力が“Low”であるか否かを判断し（ステップS42）、前入力“Low”でなければ、即ち“High”である場合（ステップS42のNOルート）、現入力“High”を前入力に置き換えて（ステップS43）、処理を終了する。

そして、ステップS42で前入力“Low”であると判断された場合（ステップS42のYESルート）、方形波発生ブロックB1からのパルスの立ち上がりエッジが検出されたことになる。この立ち上がりエッジの検出をトリガにして、制御ファームウェアとの同期処理（ステップS44～S47）へ移行する。

【0065】

この同期処理（ステップS44～S47）は、図2において説明したステップS21～S24の処理に対応するので、その説明は省略する。制御ファームウェアでサーボタスク信号がクリアされ、ステップS47でYES判定となると、同期処理を終了し、現入力“High”を前入力に置き換えて（ステップS43）、処

理を終了する。

【0066】

このとき、制御周期 ΔT とシミュレーション周期 Δt を加算して得られる合計時間 t （図4のステップS33参照）とには $\Delta t/2$ 未満の誤差を生じるが、制御周期 ΔT に対してシミュレーション周期 Δt を任意に設定することができる。なお、制御周期 ΔT がシミュレーション周期 Δt の自然数倍になっていれば前記誤差は0となるが、自然数倍になっていなければ、 $-\Delta t/2 \sim \Delta t/2$ の範囲の誤差が生じることになる。

【0067】

さらに、本実施形態の制御プログラム開発支援装置1では、シミュレーション制御部（同期処理部）22が、シミュレータ20によるシミュレーション結果に基づいて、制御回路10（制御ファームウェア）による制御量の算出動作の開始タイミングを決定するように構成することもできる。図5に示した方形波発生ブロックB1に代えて、例えば図7に示すようなサーボマーク50aの検出シミュレーション結果を同期ブロックB2に入力することにより、割込信号SVIntが一定周期ではなく他の要因に応じて生成・出力されることになる。

【0068】

なお、図7に示す例では、HDDのディスク50におけるサーボマーク50aをヘッド51によって検出する状況がシミュレータ20によりシミュレートされている。このように生成された割込信号SVIntの間隔を制御ファームウェア側で測定することにより、制御プログラム開発支援装置1は、速度（回転速度）を計測すべき処理に対応することができる。ただし、この場合、状態量の中に経過時間の情報を含ませる必要がある。

【0069】

また、本実施形態の制御プログラム開発支援装置1では、方形波発生ブロックB1から同期ブロックB2へ周期的に入力される方形波に対してわざと1パルス分の方形波を取り除く処理を加え、1パルス欠けた方形波を同期ブロックB2に入力させることにより、制御ファームウェアでのタイマによるフェイルセーフ機能がきちんと作用するか否かの検証を行なうことができる。

【 0 0 7 0 】

〔 1 - 5 〕 入出力の同期設定

次に、図 8 ～ 図 1 0 を参照しながら、本実施形態での入出力の同期設定について説明する。ここで、図 8 は本実施形態で入出力の同期設定を行なうためのモデル記述レベル（ディスプレイでの表示状態）を示す図、図 9 および図 1 0 は、それぞれ、図 8 に示すモデル記述レベルで記述・設定された同期ブロック B 2' の動作を説明するためのフローチャートおよびタイムチャートである。

【 0 0 7 1 】

図 2 に示す同期処理手順において、仮想メカモデル 2 1 は、シミュレーションを実行する前に制御回路 1 0 からの制御量を読み込み、シミュレーションを実行した後に状態量を出力しなければならない。本実施形態では、このような入出力の同期タイミングも、一般的なモデル記述方法〔例えば M A T L A B / Simulink (MathWorks 社)〕を用いて、ディスプレイ上に図 8 に示すごとく簡易なモデル（ブロック線図）をグラフィカルに記述・作成することによって設定される。

【 0 0 7 2 】

図 8 に示すモデル記述レベルにおける同期ブロック B 2' の出力は、この同期ブロック B 2' への入力パルスの立ち上がりエッジに反応し、1 シミュレーションサイクル (Δt) 分だけ “High” になる。そして、本実施形態では、図 8 に示すごとく、同期ブロック B 2' の出力が “High” になる時をトリガとして制御対象モデル B 5（シミュレータ 2 0）から状態量を出力するように出力ブロック B 4 が記述・作成される一方、同期ブロック B 2' の出力が “Low” になる時をトリガとして制御対象モデル B 5（シミュレータ 2 0）に制御量を入力するように入力ブロック B 3 が記述・作成されて、入出力タイミングが規定される。

【 0 0 7 3 】

前述した同期ブロック B 2' の動作を図 9 に示すフローチャート（ステップ S 5 1 ～ S 6 0）に従って具体的に説明する。なお、図 9 におけるステップ S 5 2 および S 5 3 は、図 6 のステップ S 4 1 および S 4 2 と同様、エッジ検出（パルス立ち上がり検出）を行なう部分である。また、ステップ S 5 7 ～ S 6 0 は、制御ファームウェアとの同期を行なう部分で、図 2 のステップ S 2 1 ～ S 2 4 に対

応する処理である。

【0074】

同期ブロック（同期処理部、シミュレーション制御部22）B2'は、シミュレーション周期 Δt 毎に起動され、その都度、まず、同期ブロックB2'からの前出力が“High”であるか否かを判断する（ステップS51）。前出力が“High”でなければ、即ち“Low”である場合（ステップS51のNOルート）、同期ブロックB2'に対する現入力（High）であるか否かを判断する（ステップS52）。現入力（High）でなければ、即ち“Low”である場合（ステップS52のNOルート）、同期ブロックB2'からの出力を“Low”としてから（ステップS54）、現入力を前入力に置き換え且つ現出力を前出力に置き換えて（ステップS56）、処理を終了する。

【0075】

現入力（High）である場合（ステップS52のYESルート）、前入力（Low）であるか否かを判断し（ステップS53）、前入力（Low）でなければ、即ち“High”である場合（ステップS53のNOルート）、前述したステップS54およびステップS56を実行して処理を終了する。

【0076】

そして、ステップS53で前入力（Low）であると判断された場合（ステップS53のYESルート）、同期ブロックB2'への入力パルスの立ち上がりエッジが検出されたことになる。この立ち上がりエッジの検出をトリガにして、同期ブロックB2'からの出力を“High”としてから（ステップS55）、現入力を前入力に置き換え且つ現出力を前出力に置き換えて（ステップS56）、処理を終了する。

【0077】

一方、ステップS51で前出力（High）であると判断された場合（ステップS51のYESルート）、制御ファームウェアとの同期処理（ステップS57～S60）へ移行する。この同期処理（ステップS57～S60）は、図2において説明したステップS21～S24の処理に対応するので、その説明は省略する。制御ファームウェアでサーボタスク信号がクリアされ、ステップS60でYE

S判定となると、同期処理を終了し、ステップS52へ移行する。

【0078】

図9にて上述した同期ステップB2'の処理について図10を参照しながら説明する。図10におけるシミュレーション時刻 t_{11} ～ t_{12} および t_{13} ～ t_{14} に示すように、同期ブロックB2'の出力は、この同期ブロックB2'への入力パルスの立ち上がりエッジに反応して（ステップS53のYESルート）、制御周期（サーボ間隔） ΔT ごとに、1シミュレーション周期（シミュレーション間隔） Δt の間だけ“High”になり（ステップS55）、それ以外の時には、同期ブロックB2'の出力は常に“Low”になる（ステップS54）。

【0079】

また、同期ブロックB2'の前出力が“High”であることをトリガとして（つまり同期ブロックB2'の出力パルスが立ち下がる直前）、シミュレータ20は、ステップS57～S60による同期処理へ移行してサーボタスク信号のクリア待ち状態になり、図10のシミュレーション時刻 t_{12} 、 t_{14} でシミュレーションを停止した状態になる。サーボタスク信号がクリアされると（ステップS60のYESルート）、シミュレータ20によるシミュレーションが再開される。

【0080】

そして、同期ブロックB2'の出力パルスの立ち下がりに応じて入力ブロックB3が機能し、制御ファームウェアからの制御量が制御対象モデルB5（仮想メカモデル21）に入力される。また、同期ブロックB2'の出力パルスの立ち上がりに応じて出力ブロックB4が機能し、制御対象モデルB5（仮想メカモデル21）の状態量が出力される。

【0081】

〔1-6〕マルチレート制御への対応手法

次に、図11～図14を参照しながら、本実施形態の制御プログラム開発支援装置1での、マルチレート制御への対応手法について説明する。ここで、図11は本実施形態でのマルチレート制御について説明するための図、図12は本実施形態でマルチレート設定を行なうためのモデル記述レベル（ディスプレイでの表示状態）を示す図、図13は本実施形態でのマルチレート制御に先立つ初期設定

手順を説明するためのフローチャート、図 1 4 は、本実施形態でのマルチレート制御手順（図 1 2 に示すモデル記述レベルで設定されたマルチレートブロック B 6 の動作）を説明するためのフローチャートである。

【0082】

制御ファームウェア（F/W）は、サーボ制御ルーチンで計算した制御量を直ちにアクチュエータに出力せず、適当な回路を用いて一定時間が経過した後に出力することもある。例えば図 1 1 に示すように、サーボ制御ルーチン（一制御周期）の間に、アクチュエータに対する出力値（制御量）を“A”から“B”に変化させる制御を行なう場合がある。このような制御はマルチレート制御と呼ばれるもので、マルチレート制御を採用することにより、制御回路 1 0（サーボ制御プログラム、制御ファームウェア）は、一制御周期中において異なるタイミングでシミュレータ 2 0（仮想メカモデル 2 1）に対して複数の制御量を出力することができる。

【0083】

本実施形態の制御プログラム開発支援装置 1 は、サーボ制御プログラムが上述のようなマルチレート制御を採用している場合にも対応することができるようになっている。そのため、制御プログラム開発支援装置 1 のシミュレータ 2 0 には、制御回路 1 0 からの複数の制御量をそれぞれ所定のタイミングで仮想メカモデル 2 1（制御対象モデル B 5）に入力するように制御量の入力制御を行なうマルチレート制御手段がそなえられる。このマルチレート制御手段は、図 1 2 に示すごとく記述・設定されるマルチレートブロック B 6 によって実現される。

【0084】

本実施形態の装置 1 をマルチレート制御に対応させるためには、図 1 1 および図 1 2 に示すごとく、ファームウェアから制御量（例えば A，B）とともにその制御量を出力するまでの時間（例えば T a，T b）を入力する。なお、その時間が固定されている場合には、仮想メカモデル 2 1 の中でその時間を指定してもよい。

【0085】

そして、本実施形態の制御プログラム開発支援装置 1 のシミュレータ 2 0 には

、上述したマルチレート制御手段を設定するための機能（マルチレート設定手段）がそなえられている。この機能は、シミュレータ 2 0 におけるグラフィカルユーザインタフェース機能（GUI 機能）を用いて実現されている。つまり、オペレータ等がシミュレータ 2 0 におけるディスプレイ（図示省略）上の表示を参照しながらキーボードやマウス等を操作することにより、マルチレート制御手段の設定が行なわれるようになっていく。本実施形態では、一般的なモデル記述方法〔例えば MATLAB/Simulink（MathWorks 社）〕を用いて、ディスプレイ上に図 1 2 に示すごとく簡易なモデル（ブロック線図）をグラフィカルに記述・作成することにより、マルチレート制御手段の設定が行なわれる。

【 0 0 8 6 】

図 1 2 に示す例では、図 8 に示したものと同様の同期ブロック B 2' のほか、入力ブロック B 3、制御対象モデル B 5、マルチレートブロック B 6 および遅延ブロック（ $1/Z$ ）B 7 が記述されている。ここで、同期ブロック B 2' の出力は、入力ブロック B 3 に入力されるとともに、遅延ブロック B 7 を介してマルチレートブロック B 6 に入力されている。

【 0 0 8 7 】

また、入力ブロック B 3 と制御対象モデル B 5 との間にマルチレートブロック B 6 が記述され、入力ブロック B 3 からのマルチレート制御にかかる制御量 A、B は、それぞれの出力タイミング（時間 T_a 、 T_b ）とともにマルチレートブロック B 6 に入力され、このマルチレートブロック B 6 から所定の出力タイミングで制御対象モデル B 5 に入力されるようになっていく。マルチレート制御以外の制御量は、入力ブロック B 3 から制御対象モデル B 5 へ直接的に入力されるようになっていく。

【 0 0 8 8 】

なお、図 1 2 において、 $1/Z$ の遅延ブロック B 7 が記述されているのは、同期ブロック B 2' からの制御パルス（出力パルス）を、1 シミュレーションサイクル Δt だけ遅延させるためである。つまり、マルチレートブロック B 6 に入力される制御パルスは、遅延ブロック B 7 により、入力ブロック B 3 に入力される制御パルスよりも 1 シミュレーションサイクル Δt だけ遅くなる。入力ブロック

B 3 およびマルチレートブロック B 6 は、いずれも、同期ブロック B 2' からの制御パルスの立ち下がりエッジをトリガとして起動される。

【 0 0 8 9 】

上述のような遅延処理を行なうことで、必ず、入力ブロック B 3 による処理が実行されてから、マルチレートブロック B 6 による処理が実行されることになる。つまり、制御回路 1 0 (ファームウェア) からの指令 (制御量, 時間) を入力ブロック B 3 で確実に受け取った後に、マルチレートブロック B 6 がマルチレート制御を実行することになる。

【 0 0 9 0 】

このとき、厳密に言えば、マルチレートブロック B 6 に $1/Z$ だけ遅れた制御パルスを与えると、マルチレート制御は Δt だけ遅延することになるが、制御周期 ΔT に対してシミュレーション周期 Δt は小さいので、上述のような Δt の遅延は無視することができる。ただし、厳密性を確保したい場合や制御周期 ΔT に対してシミュレーション周期 Δt が小さくない場合などには、マルチレートブロック B 6 において、時間 T_a や T_b から Δt を減算して用いればよい。

【 0 0 9 1 】

なお、図 1 2 では、遅延ブロック B 7 をマルチレートブロック B 6 と別個に記述しているが、マルチレートブロック B 6 の内部で、遅延ブロック B 7 と同様の処理を行なってもよく、その場合、遅延ブロック B 7 の記述を省略することができる。また、ここでは、2 段のマルチレート制御 (制御量 A, B) について説明しているが、本発明は、この段数に限定されるものではなく、同様の原理で 3 段以上のマルチレート制御にも適用される。

【 0 0 9 2 】

ついで、前述したマルチレートブロック B 6 の動作を、図 1 4 に示すフローチャート (ステップ S 7 1 ~ S 9 3) に従って具体的に説明する。

なお、マルチレート制御に先立って初期設定処理が実行されるので、まず、この初期設定処理の手順を、図 1 3 に示すフローチャート (ステップ S 6 1 ~ S 6 4) に従って説明する。つまり、シミュレータ 2 0 においては、データ用メモリを確保し (*p=new Data; ステップ S 6 1)、初期出力値 ($p \rightarrow v = 0$) を設定する

とともに（ステップ S 6 2）、初期出力時間（ $p \rightarrow t = 0$ ）を設定し（ステップ S 6 3）、さらに、リスト構造用初期値（ $p \rightarrow next = NULL$ ）を設定する（ステップ S 6 4）。

【 0 0 9 3 】

さて、図 1 4 を参照しながら、マルチレートブロック B 6 の動作について説明する。

図 1 3 や図 1 4 において、“ \rightarrow ” は、C 言語や C++ 言語の構造体もしくはクラスメンバにアクセスするための演算子を示すものである。

また、構造体もしくはクラスデータは、制御量 “ v ” と、その制御量を出力する時間 “ t ” と、さらに次のデータ へのポインタ “ $next$ ” という 3 つのメンバを有している。

【 0 0 9 4 】

さらに、変数 “ p ” は、次のような状態（データの型）で保存されている。

$p \rightarrow v$: $p \rightarrow t$ 時の出力（制御量）

$p \rightarrow t$: $p \rightarrow v$ を出力する時間

$p \rightarrow next$: 次のポインタ $p \rightarrow next \rightarrow v$

$p \rightarrow next \rightarrow t$

$p \rightarrow next \rightarrow next$: $p \rightarrow next \rightarrow next \rightarrow v$

$p \rightarrow next \rightarrow next \rightarrow t$

$p \rightarrow next \rightarrow next \rightarrow next$:

【 0 0 9 5 】

このようにして、メンバ “ $next$ ” を用いてリスト構造が形成され、 t , v の組がいくつでも記憶される。“ $next$ ” として “ $NULL$ ” を設定されたデータが、リスト構造の最後（末端）のデータである。

以下では、例えば、 $\Delta t = 1$ とし、且つ、以下のようなリスト構造のデータが設定されている場合についての、マルチレートブロック B 6 の動作を図 1 4 に従って説明する。

【 0 0 9 6 】

$p \rightarrow v = -1$

p->t = -1

p->next = P1

P1->v=2 (=p->next->v)

P1->t=1

P1->next = P2

P2->v=4 (= P1->next->v = p->next->next->v)

P2->t=3

P2->next = NULL

【 0 0 9 7 】

図 1 4 に示すシミュレーションサブルーチンは、シミュレーション周期 Δt 毎に呼び出されて実行され、まず、トリガ（遅延ブロック B 7 からの制御パルスの立ち下がリエッジ）が検出されたか否かを判断する（ステップ S 7 1）。

トリガが検出されない場合（ステップ S 7 1 の NO ルート）、下記(1)～(4)の処理が順次実行されることになる。

【 0 0 9 8 】

(1) $q \leftarrow p$, $q \rightarrow t (=p \rightarrow t) \leftarrow -2$

(2) $q \leftarrow q \rightarrow next (=P1)$, $q \rightarrow t (=P1 \rightarrow t) \leftarrow 0$

(3) $q \leftarrow q \rightarrow next (=P2)$, $q \rightarrow t (=P2 \rightarrow t) \leftarrow 2$

(4) $q \leftarrow q \rightarrow next (=NULL)$, q は NULL だからループを抜ける。

【 0 0 9 9 】

ここで、(1)は、ステップ S 7 2 の後、ステップ S 7 3 の NO ルートからステップ S 7 4 に移行した結果であり、(2)および(3)は、いずれも、ステップ S 7 5 の後、ステップ S 7 3 の NO ルートからステップ S 7 4 に移行した結果であり、(4)は、ステップ S 7 5 の後、ステップ S 7 3 の YES ルートを通して、ステップ S 7 3 ～ S 7 5 のループを抜けることを示している。

【 0 1 0 0 】

そして、“p->next” は “P1” であって “NULL” ではなく（ステップ S 7 6 の

N O ルート)、且つ、“p->next->t”は“0”であって“0”以上であるので（ステップ S 7 7 の Y E S ルート）、出力値（マルチレートブロック B 6 から出力される制御量）“p->v”として“-1”を出力し（ステップ S 8 1）、処理を終了する。この時点でのデータをまとめると、

【 0 1 0 1 】

p->v = -1

p->t = -2

p->next = P1

P1->v=2 (=p->next->v)

P1->t=0

P1->next = P2

P2->v=4 (= P1->next->v = p->next->next->v)

P2->t=2

P2->next = NULL

となる。

【 0 1 0 2 】

この後、シミュレーション周期 Δt が経過し、シミュレーションサブルーチンが呼び出されて実行され、再び、トリガが検出されない場合（ステップ S 7 1 の N O ルート）、ステップ S 7 2 ~ S 7 5 の処理により、データは、

p->v = -1

p->t = -3

p->next = P1

P1->v=2 (=p->next->v)

P1->t=-1

P1->next = P2

P2->v=4 (= P1->next->v = p->next->next->v)

P2->t=1

P2->next = NULL

となる。この後、処理は、ステップ S 7 3 の Y E S ルートを通して、ステップ S 7 3 ~ S 7 5 のループを抜けることになる。

【 0 1 0 3 】

このとき、“p->next”は“P1”であって“NULL”ではなく（ステップ S 7 6 の N O ルート）、“p->next->t”が“-1”で“0”未満であるので（ステップ S 7 7 の N O ルート）、下記(5)~(7)の処理が、それぞれ、ステップ S 7 8 ~ S 8 0 により、順次実行されることになる。

【 0 1 0 4 】

(5) q=p->next (=P1)

(6) delete p (領域 p を解放)

(7) p=q (=P1)

これに応じて、データは、

p->v =2 (=P1)

p->t =-1

p->next =P2

P2->v=4 (=p->next->v)

P2->t=1

P2->next = NULL

となり、“p->next”は“P2”であって“NULL”ではなく（ステップ S 7 6 の N O ルート）、且つ、“p->next->t”は“1”であって“0”以上であるので（ステップ S 7 7 の Y E S ルート）、出力値（マルチレートブロック B 6 から出力される制御量）“p->v”として“2”を出力し（ステップ S 8 1 ）、処理を終了する。

【 0 1 0 5 】

上述のようにして、出力値“p->v”は所定のタイミングで“-1”から“2”

に変更される。また、処理に伴い “p->next” が “NULL” となった場合（ステップ S 7 6 の Y E S ルート）には、出力値として “p->v” を出力し続けることになる。

【 0 1 0 6 】

一方、トリガが検出された場合（ステップ S 7 1 の Y E S ルート）、そのトリガに伴って、入力ブロック B 3 から新規（未処理）のマルチレートデータが入力されることがあるので、未処理の入力（マルチレートデータ）が存在するか否かを判断する（ステップ S 8 2）。

【 0 1 0 7 】

未処理の入力がない場合（ステップ S 8 2 の N O ルート）には、ステップ S 7 2 に移行して前述した処理を行なう。これに対し、未処理の入力がある場合（ステップ S 8 2 の Y E S ルート）には、ステップ S 8 3 ～ S 9 3 の処理を実行することにより、未処理のマルチレートデータを、既に時系列順に保持されているマルチレートデータ列中の適当な位置に配置し、未処理のマルチレートデータを含めたマルチレートデータ列を時系列順に保持する。

【 0 1 0 8 】

次に、上述したデータ状態に引き続いてトリガが検出され（ステップ S 7 1 の Y E S ルート）、且つ、未処理の入力（例えば v=3, t=3）があった場合（ステップ S 8 2 の Y E S ルート）について説明する。このとき、下記(8)～(13)の処理が順次実行される。

【 0 1 0 9 】

(8) まず領域を確保し、そのポインタを “tmp” とする（仮に tmp=P3 とする；ステップ S 8 3）。

(9) “tmp->v=3” , “tmp->t=3” , “tmp->next=NULL” を入力値として設定する（ステップ S 8 4）。

(10) “p->t(=-1)” よりも “tmp->t (=3)” が大きいので（ステップ S 8 5 の N O ルート）、

【 0 1 1 0 】

(11) “q = p” により、“q” として “P1” が設定される（ステップ S 8 6）。

(12) “q->next (=P2)” は存在し (ステップ S 8 7 の N O ルート)、且つ、“q->next->t (=P2->t =1)” は “tmp->t (=3)” よりも小さいので (ステップ S 8 8 の N O ルート)、“q” は “q->next (=P2)” に設定され (ステップ S 8 9)、ステップ S 8 7 に戻る。

【 0 1 1 1 】

(13) そして、今度は q->next (=P2->next =NULL) が存在しないので (ステップ S 8 7 の Y E S ルート)、“tmp-> next” として “q->next (=NULL)” を設定するとともに (ステップ S 9 0)、“q->next” として “tmp (=P3)” を設定してから (ステップ S 9 1)、ステップ S 8 2 に戻る。この後、本例では、未処理の入力はないので (ステップ S 8 2 の N O ルート)、ステップ S 7 2 に移行する。

【 0 1 1 2 】

以上の処理により、データは、

p->v =2 (=P1)

p->t =-1

p->next =P2

P2->v=4 (=p->next->v)

P2->t=1

P2->next = P3

P3->v=3 (= P2->next->v = p->next->next->v)

P3->t=3

P3->next = NULL

に更新される。

【 0 1 1 3 】

なお、ステップ S 8 5 で “p->t” よりも “tmp->t” が小さいと判断された場合 (ステップ S 8 5 の Y E S ルート) には、“tmp-> next” として “p” を設定するとともに (ステップ S 9 2)、“p” として “tmp” を設定してから (ステップ

S 9 3)、ステップ S 8 2に戻る。

以上のようにして、トリガが検出された場合、新規（未処理）の入力のすべてに対し、時間 t の大きさを比較し、“ t ”の小さい順（昇順）に入力（マルチレートデータ）を並べる処理が行なわれる。

【0 1 1 4】

〔1-7〕シミュレーションの並列処理

仮想メカモデル 2 1 が、その動作のシミュレーションを個別に実行することが可能な複数の部分から構成されるものである場合、本実施形態におけるモデル実行環境（シミュレータ）2 0 を、前記複数の部分それぞれの動作を並列的にシミュレートする複数のプロセッサ（図 1 5 のMCU 1 2 a ~ 1 2 c 参照）によって構成することで、シミュレーション処理の高速化をはかることができる。

【0 1 1 5】

例えば制御対象（仮想メカモデル 2 1）がHDDである場合、本実施形態におけるモデル実行環境（シミュレータ）2 0 においては、図 1 5 に示すように、仮想メカモデル 2 1 をディスクモデル、アームモデル、流体モデルのような相関関係の低い3つの部分に分割し、各部分の動作を個別のプロセッサ（例えばMCU 1 2 a, 1 2 b, 1 2 c）により並列的に解析してシミュレートすることができる。これにより、シミュレーション処理を大幅に高速化することができる。なお、図 1 5 は本実施形態でのシミュレーションの並列処理を説明するための図である。

【0 1 1 6】

この場合、これらのMCU 1 2 a ~ 1 2 c のうちの 하나가 マスタとなり、その他のスレーブMCUは、マスタが発生する信号に対し割込やポーリングを行なうことによって、これらのMCU 1 2 a ~ 1 2 c のシミュレーション時間を同期させることができる。また、同期をとる間隔は、アームとディスクのように相関関係が低いモデルどうしについてはサーボ間隔（制御周期 ΔT ）とし、また流体が絡むなど、多少の相関関係があるモデルどうしについてはシミュレーション間隔（シミュレーション周期 Δt ）とする。

【0 1 1 7】

〔 1 - 8 〕 中継回路の詳細構成および各種機能

図 1 6 は本実施形態の中継回路 3 0 の構成および割込信号 S V I n t の取扱を説明するためのブロック図である。

本実施形態の中継回路 3 0 は、図 1 にて前述したごとく、2 つのプロセッサ（制御回路 1 0 およびシミュレータ 2 0）の間を繋ぐバス上に共有メモリ 3 1 をそなえて構成されている。より詳細に説明すると、中継回路 3 0 は、図 1 6 に示すように、共有メモリ 3 1 を成す複数のレジスタ 3 1 a と、第 1 書込／読出制御部として機能するセクタ 3 2 と、第 2 書込／読出制御部として機能するセクタ 3 3 とをそなえて構成されている。

【 0 1 1 8 】

ここで、レジスタ 3 1 a は、それぞれ、制御回路 1 0（制御ファームウェア）からシミュレータ 2 0 への制御量とシミュレータ 2 0 から制御回路 1 0（制御ファームウェア）への状態量とを含むデータを一時的に保持しうるものである。

セクタ 3 2 は、複数のレジスタ 3 1 a と制御回路 1 0 との間でデータの書込／読出を制御するもので、制御回路 1 0 から F / W アドレスバス 6 1 を介して与えられたアドレス情報に応じて、複数のレジスタ 3 1 a のうちの一つ（アドレス情報に応じたレジスタ 3 1 a）と F / W データバス 6 2 とを接続するように切替動作を行なうものである。

【 0 1 1 9 】

また、セクタ 3 3 は、複数のレジスタ 3 1 a とシミュレータ 2 0 との間でデータの書込／読出を制御するもので、シミュレータ 2 0 からシミュレータアドレスバス 7 1 を介して与えられたアドレス情報に応じて、複数のレジスタ 3 1 a のうちの一つ（アドレス情報に応じたレジスタ 3 1 a）とシミュレータデータバス 7 2 とを接続するように切替動作を行なうものである。

【 0 1 2 0 】

このような構成により、制御回路 1 0（ファームウェア）からの制御量は、データバス 6 2 およびセクタ 3 2 を通じ、指定されたレジスタ 3 1 a に一旦書き込まれた後、同期信号（サーボタスク信号のクリア）に応じて、セクタ 3 3 およびデータバス 7 2 を通じシミュレータ 2 0 に入力される。一方、シミュレータ

20からの状態量は、データバス72およびセレクタ33を通じ、指定されたレジスタ31aに一旦書き込まれた後、同期信号（割込信号SVInt）に応じて、セレクタ32およびデータバス62を通じて制御回路10に入力される。

【0121】

このとき、本実施形態の中継回路30では、シミュレータ20からの割込信号SVIntが、特定のレジスタ31a（図16では最上段のレジスタ）における所定ビットnに書き込まれ、そのビットnの値（割込信号SVInt）が、直接外部に出力され、セレクタ32を介することなくレジスタ31aから制御回路10へ直接的に送出されるように構成されている。

【0122】

これにより、制御回路10側では、セレクタ32による読出制御（ファームウェアの読み込み動作）を行なうことなく割込信号SVIntを得ることができ、ハードウェア割り込みを利用した同期処理を行なうことが可能になる。従って、ファームウェアのメインループにてSVIntを常に監視するためのポーリングを行なう必要がなくなる。

【0123】

なお、図16に示す中継回路30の共有メモリ31は、双方向アクセスが可能なレジスタ31aにより構成されているが、回路を簡素化するために、制御ファームウェアがライト専用で用い且つシミュレータ20がリード専用で用いるレジスタと、シミュレータ20がライト専用で用い且つ制御ファームウェアがリード専用で用いるレジスタとの二種類から構成されてもよい。さらに、レジスタ31aの一方の側だけリード／ライト可能な構成としてもよい。この場合、割込信号SVIntは、シミュレータ20側からライト可能なレジスタ31aにおける所定ビットに書き込まれる。

【0124】

ところで、本発明の目的は、ファームウェア（サーボ制御プログラム）の開発・デバッグ（検証）を支援することであるが、そのためには、制御回路10とシミュレータ20との間における通信内容を傍受したり、その通信内容を捏造したりすることができると、デバッグ時に便利である。

そこで、本実施形態では、図 1 7 ～ 図 2 0 に示すような各種機能が中継機構 3 0 にそなえられている。

【 0 1 2 5 】

ここで、図 1 7 は任意のレジスタ 3 1 a の内容を表示する機能をそなえた中継回路 3 0 の構成を示すブロック図、図 1 8 は特定のレジスタ 3 1 a の内容を表示する機能をそなえた中継回路 3 0 の構成を示すブロック図、図 1 9 は特定のレジスタ 3 1 a にデータを設定する機能をそなえた中継回路 3 0 の構成を示すブロック図、図 2 0 はレジスタ 3 1 a からのデータにノイズを重畳する機能をそなえた中継回路 3 0 の構成を示すブロック図である。

【 0 1 2 6 】

図 1 7 に示す中継回路 3 0 では、複数のレジスタ 3 1 a に保持されているデータを表示しうるデータ表示用セグメント（データ表示部） 3 6 がそなえられるとともに、このセグメント 3 6 に表示すべきデータを保持するレジスタ 3 1 a を選択・指定するための、レジスタ選択スイッチ（選択部） 3 4 およびセクタ（選択部） 3 5 がそなえられている。

【 0 1 2 7 】

レジスタ選択スイッチ 3 4 は、例えばディップスイッチ、ロータリスイッチ等で構成されオペレータ等により手動操作されるものである。また、セクタ 3 5 は、レジスタ選択スイッチ 3 4 からの信号に応じ、複数のレジスタ 3 1 a のうちのひとつ（前記信号に応じたレジスタ 3 1 a）とセグメント 3 6 とを接続するように切替動作を行ない、そのレジスタ 3 1 a に保持されているデータをセグメント 3 6 に表示させるものである。

【 0 1 2 8 】

これにより、制御回路 1 0 とシミュレータ 2 0 との間で通信中のデータであって任意のレジスタ 3 1 a に保持されるデータが、オペレータ等の指示に応じてセグメント 3 6 で表示され、オペレータ等はそのデータを参照・確認することができる。

なお、ここで、回路規模を小さくするため、セクタ 3 5 として、バス用セクタ 3 2 もしくは 3 3 を流用し、バスの未使用時にのみ、そのセクタ 3 2 もし

くは 3 3 を介してセグメント 3 6 にデータを表示するようにしてもよい。

【 0 1 2 9 】

また、図 1 7 に示すごとくセグメント 3 6 でデジタル表示する他に、セレクタ 3 5 で選択されたデータを、そのままデジタル信号として出力し他のコンピュータに入力して記録したり、D/A 変換器を介してアナログ出力しオシロスコープなどで表示・観察したりしてもよい。

【 0 1 3 0 】

さらに、図 1 8 に示すごとく、スイッチ 3 4 やセレクタ 3 5 を省略し、データ表示用セグメント 3 6 a で表示するレジスタ 3 1 a を特定のものに固定してもよい。つまり、図 1 8 に示す中継回路 3 0 では、セグメント 3 6 a は、複数のレジスタ 3 1 a のうちの特定のものに直接的に接続され、その特定のレジスタ 3 1 a に保持されているデータのみをデジタル表示している。

【 0 1 3 1 】

また、図 1 9 に示す中継回路 3 0 では、複数のレジスタ 3 1 a のうちの少なくとも一つに所望のデータを強制的に設定・格納するためのスイッチ（データ入力部） 3 7 がそなえられている。

本実施形態において、スイッチ 3 7 は、例えばディップスイッチ、ロータリスイッチ等で構成されオペレータ等により手動操作されるもので、複数のレジスタ 3 1 a のうちの特定のものに直接的に接続され、その特定のレジスタ 3 1 a に、スイッチ 3 7 によって指定されたデータが設定入力されるようになっている。このスイッチ 3 7 を接続されたレジスタ 3 1 a は、読み取り専用となっている。

【 0 1 3 2 】

そして、このように特定のレジスタ 3 1 a に設定されたデータを、ファームウェア（あるいはシミュレータ 2 0）で読めるようにすることにより、そのデータに応じた制御プログラムの挙動や仮想メカモデル 2 1 の動作状態を確認することができ、制御プログラムのデバッグを支援することができる。

【 0 1 3 3 】

なお、スイッチ 3 7 からデータを手動入力する他に、外部から与えられたアナログデータを A/D 変換器によりデジタルデータに変換して特定のレジスタ 3 1

aに設定入力したり、他のコンピュータの出力であるデジタルデータを、直接、特定のレジスタ31aに設定入力したりすることで、デバッグを支援することも考えられる。

【0134】

また、図19に示すような機能を用いることにより、サーボゲインの調整を手動入力によりリアルタイムで行なったり、FFT (Fast Fourier Transform) アナライザを用いて解析を行なったりすることができる。

FFTアナライザを用いる場合、FFTアナライザはアナログ信号を入出力するものであるため、FFTアナライザからの出力（アナログ信号）を、A/D変換器によりデジタル信号に変換して特定のレジスタ31aに設定し、対象物（シミュレータ20もしくはファームウェア）からの応答を待つ。そして、対象物からの応答（デジタル信号）をD/A変換器によりアナログ信号に変換してからFFTアナライザに入力することになる。

【0135】

さらに、図20に示す中継回路30では、特定のレジスタ31aから読み出されたデータにスイッチ39を介してノイズを重畳する加算器（ノイズ重畳部）38a、38bが、レジスタ31aの両側（ファームウェア側およびシミュレータ20側）のそれぞれそなえられている。

これにより、オペレータ等は、ノイズを加算器38a、38bによってデータに重畳することができ、そのノイズに応じたファームウェア（制御プログラム）の挙動やモデルの動作状態を確認することができる。

【0136】

なお、図20に示す例では、加算器38a、38bを用いて、外部からのノイズを特定のレジスタ31a（アドレス）に加えているが、加算器38a、38bの代わりに乗算器を用いてもよい。また、図20では、レジスタ31aの両側のバスに加算器38a、38bをそなえているが、いずれか一方のバスのみに加算器をそなえてもよい。その他、セレクタ（図示省略）を用い任意のレジスタからの読み込み値に対してノイズを加算する機能を付加してもよい。

【0137】

〔 1 - 9 〕 本実施形態の効果

このように、本発明の一実施形態としての制御プログラム開発支援装置 1 によれば、制御回路 1 0 (M P U 1 2) での演算処理を遅らせてシミュレータ 2 0 でのモデル演算処理 (シミュレーション) と同期を取ることにより、サーボ特性を変化させずにスローモーション的に且つ時間厳密性を保ったまま、精密なシミュレーションが行なわれる。従って、実際のメカを用いることなく、比較的小型で応答の速い製品についてのサーボ制御プログラムの開発・デバッグ (検証) を行なうことができる。

【 0 1 3 8 】

また、モデルパラメータを変更するだけで容易に任意の特性をもった仮想メカモデル 2 1 を作成してサーボ制御プログラム (ファームウェア) によって制御させることができるので、サーボ制御プログラムが、大量に生産される製品のバラツキにどの程度まで対応できるかの検証、つまりサーボ制御プログラムの品質検証を確実に行なうことができる。また、任意のタイミングで任意の外乱を与えることが可能なため、サーボ制御プログラムの品質をより確実に検証することができる。

【 0 1 3 9 】

さらに、仮想メカモデル 2 1 を用いてシミュレーションを行なうことにより、イベントブレイクやステップデバッグなどの機能を使用することが可能になり、サーボ制御プログラム開発をより容易に行なえるほか、新しいアクチュエータやセンサを用いた新規の制御手法も簡単に検証することが可能になる。

オペレータ等は、図 5 や図 8 で説明した同期設定手段 (G U I 機能) を用いることにより、シミュレータ 2 0 の動作と制御回路 1 0 (ファームウェア) の動作との同期設定を容易かつ任意に行なうことができる。

【 0 1 4 0 】

図 8 ～図 1 0 で説明したマルチレート制御手段を用いて、一制御周期中に制御量が増加するマルチレート制御がシミュレートされるので、本実施形態の制御プログラム開発支援装置 1 は、サーボロジックがマルチレート制御を実現していた場合に確実に対応することができる。その際、オペレータ等は、図 8 で説明した

マルチレート設定手段（GUI機能）を用いることにより、そのマルチレート制御を容易かつ任意に定義・設定することができる。

【0141】

図7にて説明したように、シミュレータ20によるシミュレーション結果に応じて、サーボ制御ルーチンへ移行することができるので、タイマによるフェイルセーフ機能の確認や、単位時間当たりの変化量（速度、回転数等）を測定する処理への対応など、各種機能が実現され、サーボ制御プログラムの開発・デバッグ（検証）を確実に支援することができる。

【0142】

また、図15にて説明したように、仮想メカモデル21の構成部分の動作シミュレーションを、複数のプロセッサ（本実施形態ではMCU12a～12c）で並列的に実行することができるので、シミュレーション処理を大幅に高速化することができる。

【0143】

さらに、本実施形態では、図16～図20に示すように、中継回路30を、複数のレジスタ31a，セクタ32および33によって構成することで、制御回路10からの制御量やシミュレータ20からの状態量を、レジスタ31aにおいて一時的に保持してから、シミュレータ20や制御回路10に確実に中継することができる。

【0144】

このとき、ファームウェア（制御回路10）側ではセクタ32による読出制御を行なうことなく割込信号SVIntを得ることができるので、ハードウェア割り込みを利用した同期処理を行なうことが可能になり、制御回路10は、その割込信号SVIntに応じて、制御量の算出動作を直ちに且つ確実に開始することができる。

【0145】

また、図17や図18に示すごとく、制御回路10とシミュレータ20との間で通信中のデータを表示するデータ表示用セグメント36，36aにより、オペレータ等はそのデータを参照・確認することができるので、サーボ制御プログラ

ムの開発・デバッグ（検証）を確実に支援することができる。

【0146】

さらに、図19に示すごとく、オペレータ等は、任意のデータを、スイッチ37からレジスタ31aに書き込むことによって制御回路10やシミュレータ20へ直接的に入力することができるので、そのデータに応じたサーボ制御プログラムの挙動や仮想メカモデル21の動作状態を確認することが可能になり、サーボ制御プログラムの開発・デバッグ（検証）を確実に支援することができる。

【0147】

そして、図20に示すごとく、オペレータ等は、ノイズを加算器38a, 38bによってデータに重畳することができるので、そのノイズに応じたサーボ制御プログラムの挙動やモデルの動作状態を確認することが可能になり、サーボ制御プログラムの開発・デバッグ（検証）を確実に支援することができる。

【0148】

〔2〕その他

なお、本発明は上述した実施形態に限定されるものではなく、本発明の趣旨を逸脱しない範囲で種々変形して実施することができる。

例えば、上述した実施形態では、制御対象が磁気ディスクドライブ（HDD）である場合について説明したが、本発明は、これに限定されるものではなく、光ディスク（CD, MO, DVD, MD）、磁気テープ装置（DAT, VTR）、NC工作機など、緻密なサーボ制御を必要とするあらゆる分野に応用することができる。さらに、上述した実施形態では、制御対象がサーボ機構である場合について説明したが、本発明は、このようなサーボ機構に限定されるものではない。

【0149】

〔3〕付記

（付記1） 機構の動作を制御する制御プログラムを実行し、該機構に対する制御量を所定の制御周期で算出して出力する制御プログラム実行部と、

該機構を仮想的なモデルとして内部に構築され、該モデルを用い、前記所定制御周期よりも短く設定された所定のシミュレーション周期で、前記所定の制御周期に対応する時間に亘って該機構の動作をシミュレートすることにより、該機構

の状態量を算出して出力するシミュレーション部と、

該制御プログラム実行部から該シミュレーション部への前記制御量、および、該シミュレーション部から該制御プログラム実行部への前記状態量を一時的に保持し中継する中継部と、

該シミュレーション部からの前記状態量が該中継部に保持されると、該シミュレーション部を、該制御プログラム実行部からの応答待ち状態へ移行させるとともに、該制御プログラム実行部による、前記状態量に応じた制御量の算出動作を開始させる一方、該制御プログラム実行部からの前記制御量が該中継部に保持されると、該制御プログラム実行部を、該シミュレーション部からの応答待ち状態へ移行させるとともに、該シミュレーション部による、前記制御量に応じたシミュレーション動作を開始させるシミュレーション制御部とをそなえたことを特徴とする、制御プログラム開発支援装置。

【 0 1 5 0 】

（付記 2） サーボ機構の動作を制御する制御プログラムを実行し、該サーボ機構に対する制御量を所定の制御周期で算出して出力する制御プログラム実行部と、

該サーボ機構を仮想的なモデルとして内部に構築され、該モデルを用いて該サーボ機構の動作を動力学的に解析しながら、前記所定制御周期よりも短く設定された所定のシミュレーション周期で、前記所定の制御周期に対応する時間に亘って該サーボ機構の動作をシミュレートすることにより、該サーボ機構の状態量を算出して出力するシミュレーション部と、

該制御プログラム実行部から該シミュレーション部への前記制御量、および、該シミュレーション部から該制御プログラム実行部への前記状態量を一時的に保持し中継する中継部と、

該シミュレーション部からの前記状態量が該中継部に保持されると、該シミュレーション部を、該制御プログラム実行部からの応答待ち状態へ移行させるとともに、該制御プログラム実行部による、前記状態量に応じた制御量の算出動作を開始させる一方、該制御プログラム実行部からの前記制御量が該中継部に保持されると、該制御プログラム実行部を、該シミュレーション部からの応答待ち状態

へ移行させるとともに、該シミュレーション部による、前記制御量に応じたシミュレーション動作を開始させるシミュレーション制御部とをそなえたことを特徴とする、制御プログラム開発支援装置。

【 0 1 5 1 】

（付記 3） 該シミュレーション制御部の同期設定を行なうための同期設定手段をそなえたことを特徴とする、付記 2 記載のサーボ制御プログラム開発支援装置。

（付記 4） 該同期設定手段が、グラフィカルユーザインタフェース機能を用いて構成されていることを特徴とする、付記 3 記載の制御プログラム開発支援装置。

【 0 1 5 2 】

（付記 5） 該制御プログラム実行部が、一制御周期中において異なるタイミングで該シミュレーション部に入力されるべき複数の制御量を出力するものであり、

前記複数の制御量をそれぞれ所定のタイミングで該シミュレーション部に入力するように制御量の入力制御を行なうマルチレート制御手段をそなえたことを特徴とする、付記 1 記載の制御プログラム開発支援装置。

【 0 1 5 3 】

（付記 6） 該マルチレート制御手段の設定を行なうためのマルチレート設定手段をそなえたことを特徴とする、付記 5 記載の制御プログラム開発支援装置。

（付記 7） 該マルチレート設定手段が、グラフィカルユーザインタフェース機能を用いて構成されていることを特徴とする、付記 6 記載の制御プログラム開発支援装置。

【 0 1 5 4 】

（付記 8） 該シミュレーション制御部が、該シミュレーション部によるシミュレーション結果に基づいて、該制御プログラム実行部による前記制御量の算出動作の開始タイミングを決定することを特徴とする、付記 2 記載の制御プログラム開発支援装置。

（付記 9） 該モデルが、その動作のシミュレーションを個別に実行すること

が可能な複数の部分から構成されるものであり、

該シミュレーション部が、前記複数の部分それぞれの動作を並列的にシミュレートする複数のプロセッサをそなえて構成されていることを特徴とする、付記 2 記載の制御プログラム開発支援装置。

【 0 1 5 5 】

(付記 1 0) 該中継部が、

該制御プログラム実行部から該シミュレーション部への前記制御量と該シミュレーション部から該制御プログラム実行部への前記状態量とを含むデータを一時的に保持しうる複数のレジスタと、

該複数のレジスタと該制御プログラム実行部との間で前記データの書込／読出を制御する第 1 書込／読出制御部と、

該複数のレジスタと該シミュレーション部との間で前記データの書込／読出を制御する第 2 書込／読出制御部とをそなえて構成されていることを特徴とする、付記 2 記載の制御プログラム開発支援装置。

【 0 1 5 6 】

(付記 1 1) 該制御プログラム実行部による前記制御量の算出動作を開始させるべく該シミュレーション部から該複数のレジスタの一つに入力された割込信号については、該第 1 書込／読出制御部を介することなく、当該レジスタから該制御プログラム実行部へ直接的に送出されることを特徴とする、請求項 1 0 記載の制御プログラム開発支援装置。

【 0 1 5 7 】

(付記 1 2) 該複数のレジスタに保持されているデータを表示しうるデータ表示部をそなえたことを特徴とする、付記 1 0 記載の制御プログラム開発支援装置。

(付記 1 3) 該複数のレジスタの中から選択した、少なくとも一つのレジスタに保持されているデータを該データ表示部に表示させる選択部をそなえたことを特徴とする、付記 1 2 記載の制御プログラム開発支援装置。

【 0 1 5 8 】

(付記 1 4) 該データ表示部が、該複数のレジスタのうちの特定のものに直

接的に接続され、該特定のレジスタに保持されているデータを表示することを特徴とする、付記 1 2 記載の制御プログラム開発支援装置。

(付記 1 5) 該複数のレジスタのうちの少なくとも一つに所望のデータを強制的に設定・格納するためのデータ入力部をそなえたことを特徴とする、付記 1 0 記載の制御プログラム開発支援装置。

【0 1 5 9】

(付記 1 6) 該データ入力部が、該複数のレジスタのうちの特定のものに直接的に接続され、該特定のレジスタに前記所望のデータを設定することを特徴とする、付記 1 5 記載の制御プログラム開発支援装置。

(付記 1 7) 該複数のレジスタのうちの少なくとも一つから読み出されたデータにノイズを重畳するノイズ重畳部をそなえたことを特徴とする、付記 1 0 記載の制御プログラム開発支援装置。

【0 1 6 0】

【発明の効果】

以上詳述したように、本発明の制御プログラム開発支援装置によれば、以下のような効果ないし利点を得ることができる。

(1) 制御プログラム実行部での演算処理を遅らせてシミュレーション部でのモデル演算処理(シミュレーション)と同期を取ることで、機構の特性(サーボ特性)を変化させずにスローモーション的に且つ時間厳密性を保ったまま、精密なシミュレーションが行なわれる。従って、実際のメカを用いることなく、比較的小型で応答の速い製品についての制御プログラムの開発・デバッグ(検証)を行なうことができる(請求項 1, 2)。

【0 1 6 1】

(2) モデルパラメータを変更するだけで容易に任意の特性をもったモデルを作成して制御プログラムによって制御させることができる。従って、制御プログラムが、大量に生産される製品のバラツキにどの程度まで対応できるかの検証、つまり制御プログラムの品質検証を確実に行なうことができる(請求項 1, 2)。

【0 1 6 2】

(3) 仮想的なモデルを用いてシミュレーションを行なうことにより、ステップデバッグなどの機能を使用することが可能になり、制御プログラム開発をより容易に行なえるほか、新しいアクチュエータやセンサを用いた新規の制御手法も簡単に検証することが可能になる（請求項1，2）。

(4) オペレータ等は、同期設定手段（GUI機能）を用いて、シミュレーション部の動作と制御プログラム実行部の動作との同期設定を容易かつ任意に行なうことができる。

【0163】

(5) マルチレート制御手段を用いて、一制御周期中に制御量が増減するマルチレート制御がシミュレートされるので、サーボロジックがマルチレート制御を実現していた場合に確実に対応することができる（請求項3）。その際、オペレータ等は、マルチレート設定手段（GUI機能）を用いて、そのマルチレート制御を容易かつ任意に定義・設定することができる。

【0164】

(6) シミュレーション部によるシミュレーション結果に応じて、制御プログラム実行部での制御ルーチンへ移行することができるので、タイマによるフェイルセーフ機能の確認や、単位時間当たりの変化量（速度、回転数等）を測定する処理への対応など、各種機能が実現され、制御プログラムの開発・デバッグ（検証）を確実に支援することができる（請求項4）。

【0165】

(7) モデルの構成部分の動作シミュレーションを、複数のプロセッサで並列的に実行することができるので、シミュレーション処理を大幅に高速化することができる（請求項5）。

(8) 中継部を、複数のレジスタ、第1書込／読出制御部および第2書込／読出制御部によって構成することで、制御プログラム実行部からの制御量やシミュレーション部からの状態量を、レジスタにおいて一時的に保持してから、シミュレーション部や制御プログラム実行部に確実に中継することができる。

【0166】

(9) 制御プログラム実行部側では第1書込／読出制御部による読出制御を行

なうことなく割込信号を得ることができるので、ハードウェア割り込みを利用した同期処理を行なうことが可能になり、制御プログラム実行部は、その割込信号に応じて、制御量の算出動作を直ちに且つ確実に開始することができる。

(10) 制御プログラム実行部とシミュレーション部との間で通信中のデータを表示するデータ表示部により、オペレータ等はそのデータを参照・確認することができるので、制御プログラムの開発・デバッグ(検証)を確実に支援することができる。

【0167】

(11) オペレータ等は、任意のデータを、データ入力部からレジスタに書き込むことによって制御プログラム実行部やシミュレーション部へ直接的に入力することができるので、そのデータに応じた制御プログラムの挙動やモデルの動作状態を確認することが可能になり、制御プログラムの開発・デバッグ(検証)を確実に支援することができる。

【0168】

(12) オペレータ等は、ノイズをノイズ重畳部によってデータに重畳することができるので、そのノイズに応じた制御プログラムの挙動やモデルの動作状態を確認することが可能になり、制御プログラムの開発・デバッグ(検証)を確実に支援することができる。

【図面の簡単な説明】

【図1】

本発明の一実施形態としての制御プログラム開発支援装置の構成を従来システムの構成と比較して示すもので、(a)は従来システムの構成を示すブロック図、(b)は本実施形態の構成を示すブロック図である。

【図2】

本実施形態での同期処理手順(シミュレーション制御部の動作)を説明するためのフローチャートである。

【図3】

本実施形態での同期処理手順(シミュレーション制御部の動作)を説明するためのタイムチャートである。

【図 4】

本実施形態のシミュレータでのシミュレーション原理を説明するためのフローチャートである。

【図 5】

本実施形態で同期設定を行なうためのモデル記述レベル（ディスプレイでの表示状態）を示す図である。

【図 6】

図 5 に示すモデル記述レベルで記述・設定された同期ブロックの動作を説明するためのフローチャートである。

【図 7】

本実施形態の装置において、割り込み間隔に応じ速度を計測する手法について説明するための図である。

【図 8】

本実施形態で入出力の同期設定を行なうためのモデル記述レベル（ディスプレイでの表示状態）を示す図である。

【図 9】

図 8 に示すモデル記述レベルで記述・設定された同期ブロックの動作を説明するためのフローチャートである。

【図 10】

図 8 に示すモデル記述レベルで記述・設定された同期ブロックの動作を説明するためのタイムチャートである。

【図 11】

本実施形態でのマルチレート制御について説明するための図である。

【図 12】

本実施形態でマルチレート設定を行なうためのモデル記述レベル（ディスプレイでの表示状態）を示す図である。

【図 13】

本実施形態でのマルチレート制御に先立つ初期設定手順を説明するためのフローチャートである。

【図 1 4】

本実施形態でのマルチレート制御手順（図 1 2 に示すモデル記述レベルで設定されたマルチレートブロックの動作）を説明するためのフローチャートである。

【図 1 5】

本実施形態でのシミュレーションの並列処理を説明するための図である。

【図 1 6】

本実施形態の中継回路の構成および割込信号の取扱を説明するためのブロック図である。

【図 1 7】

任意のレジスタの内容を表示する機能をそなえた中継回路の構成を示すブロック図である。

【図 1 8】

特定のレジスタの内容を表示する機能をそなえた中継回路の構成を示すブロック図である。

【図 1 9】

特定のレジスタにデータを設定する機能をそなえた中継回路の構成を示すブロック図である。

【図 2 0】

レジスタからのデータにノイズを重畳する機能をそなえた中継回路の構成を示すブロック図である。

【符号の説明】

- 1 制御プログラム開発支援装置
 - 1 0 制御回路（制御プログラム実行部，ファームウェア実行用プロセッサ，制御ファームウェア）
 - 1 1 制御用 L S I
 - 1 2，1 2 a，1 2 b，1 2 c M C U （プロセッサ）
 - 1 3 メモリ
 - 2 0 モデル実行環境（シミュレーション部，モデル演算用プロセッサ，シミュレータ）

- 2 1 仮想メカモデル
- 2 2 シミュレーション制御部（同期処理部）
- 3 0 中継回路（中継部）
- 3 1 共有メモリ（バッファ、レジスタ）
- 3 1 a レジスタ
- 3 2 セレクタ（第 1 書込／読出制御部）
- 3 3 セレクタ（第 2 書込／読出制御部）
- 3 4 レジスタ選択スイッチ（選択部）
- 3 5 セレクタ（選択部）
- 3 6, 3 6 a データ表示用セグメント（データ表示部）
- 3 7 スイッチ（データ入力部）
- 3 8 a, 3 8 b 加算器（ノイズ重畳部）
- 3 9 スイッチ
- 4 0 状態表示記録部（データ表示部）
- 5 0 ディスク
- 5 0 a サーボマーク
- 5 1 ヘッド
- 6 1 F／Wアドレスバス
- 6 2 F／Wデータバス
- 7 1 シミュレータアドレスバス
- 7 2 シミュレータデータバス
- 1 0 0 メカ
- 1 1 0 アクチュエータ
- 1 2 0 センサ
- 2 0 0 制御回路
- 2 1 0 制御用 L S I
- 2 1 1 C P U
- 2 1 2 メモリ
- 2 1 3 サーボロジック

2 1 4 センサロジック

2 2 0 ドライバ

2 3 0 検出回路

3 0 0 状態表示部

B 2, B 2' 同期ブロック (同期処理部, シミュレーション制御部)

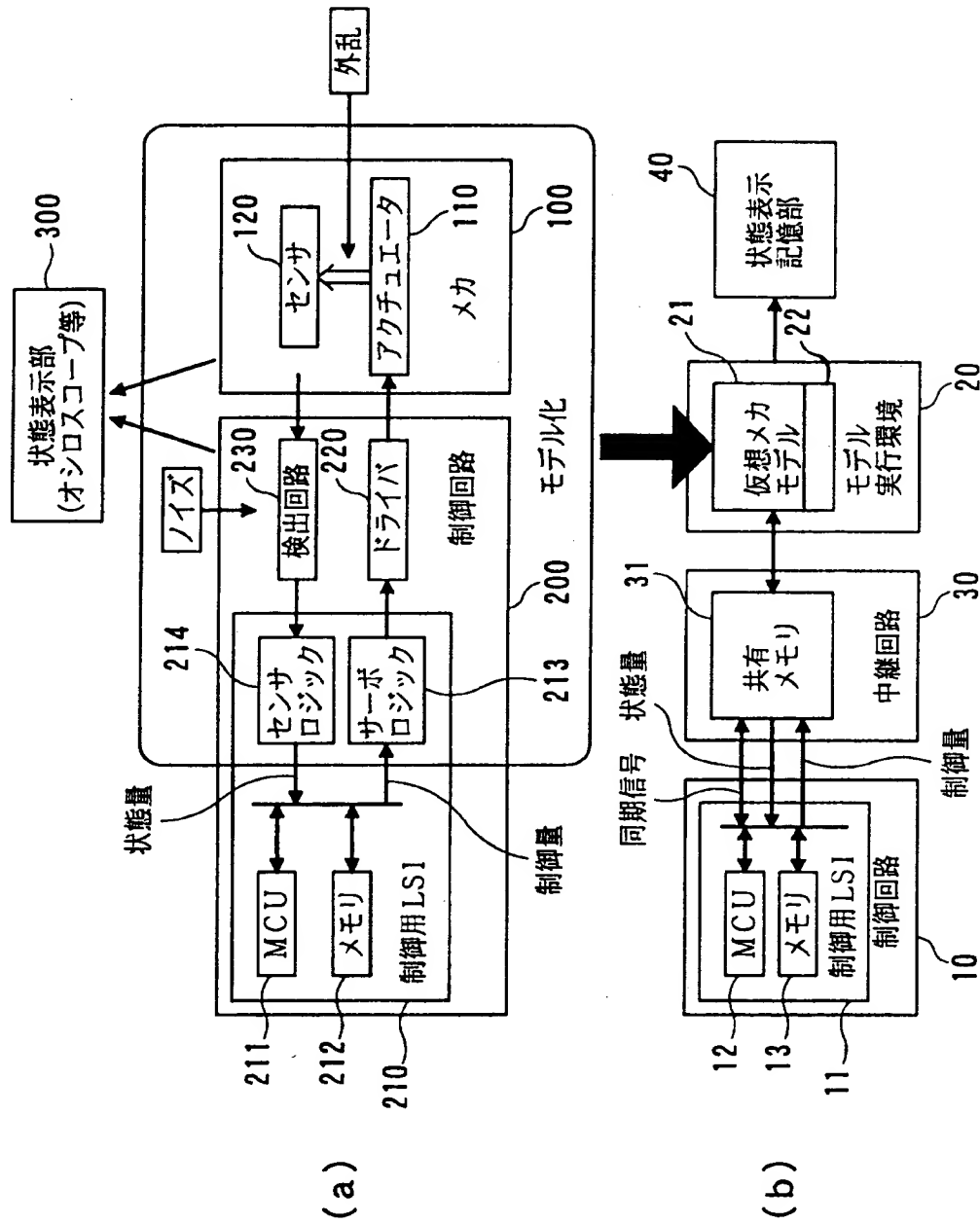
B 5 制御対象モデル

B 6 マルチレートブロック (マルチレート制御手段)

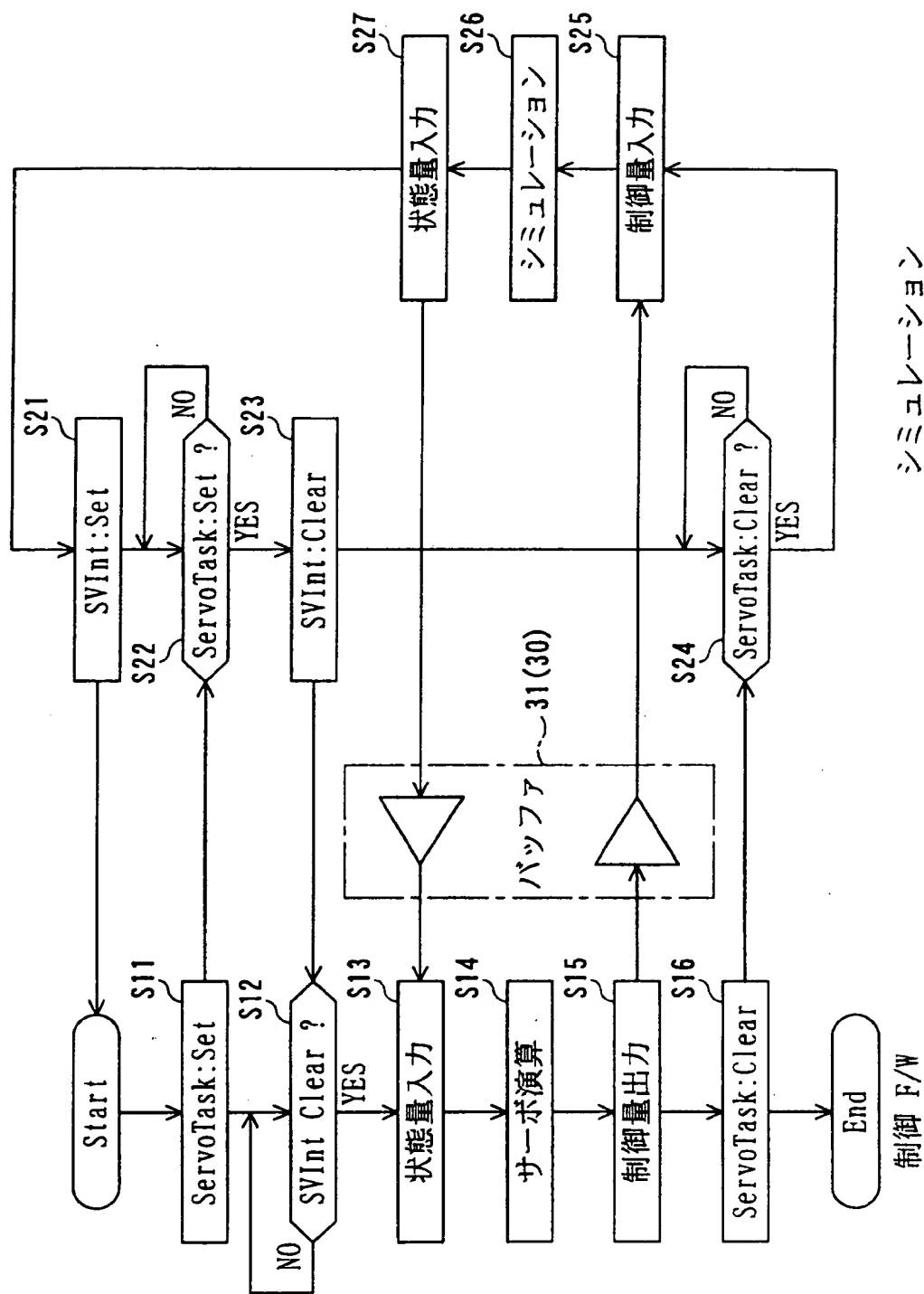
【書類名】

図面

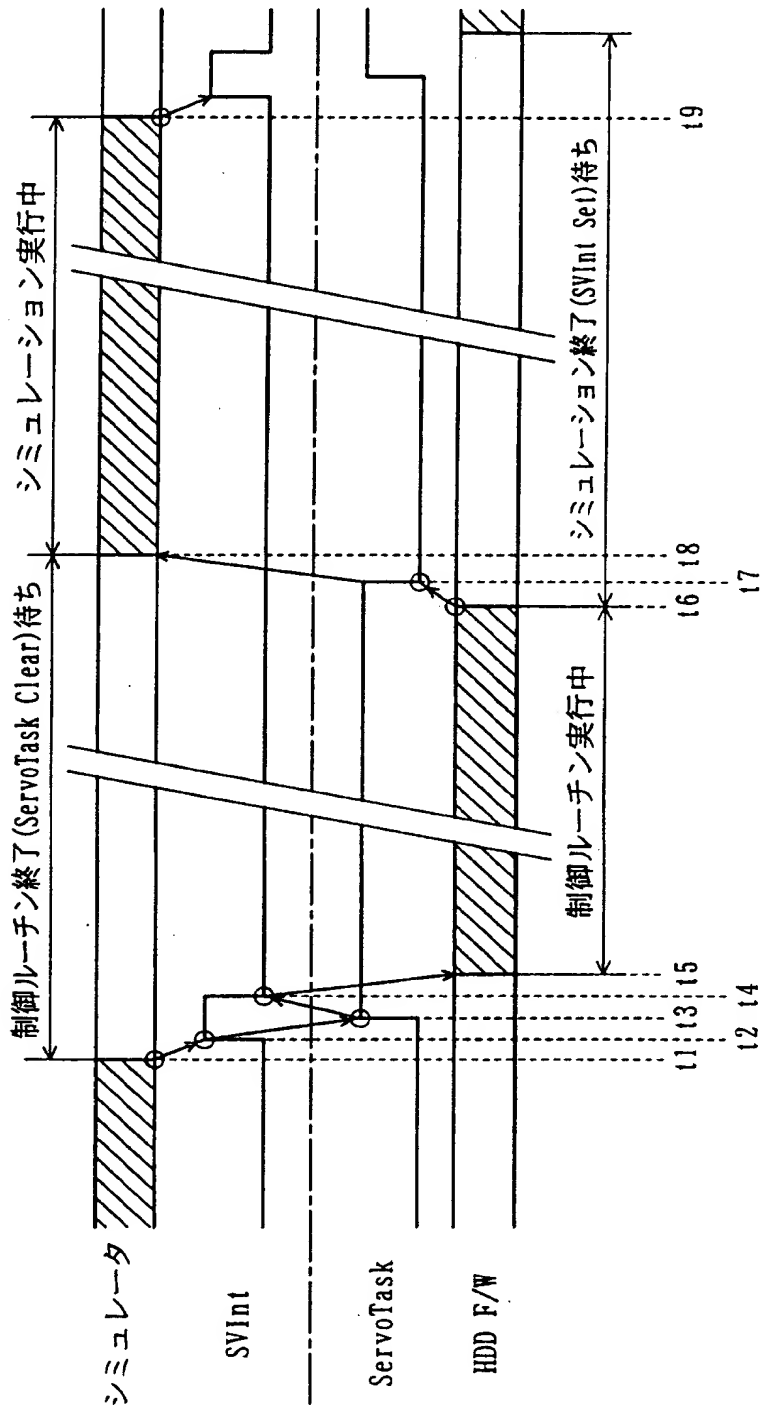
【図 1】



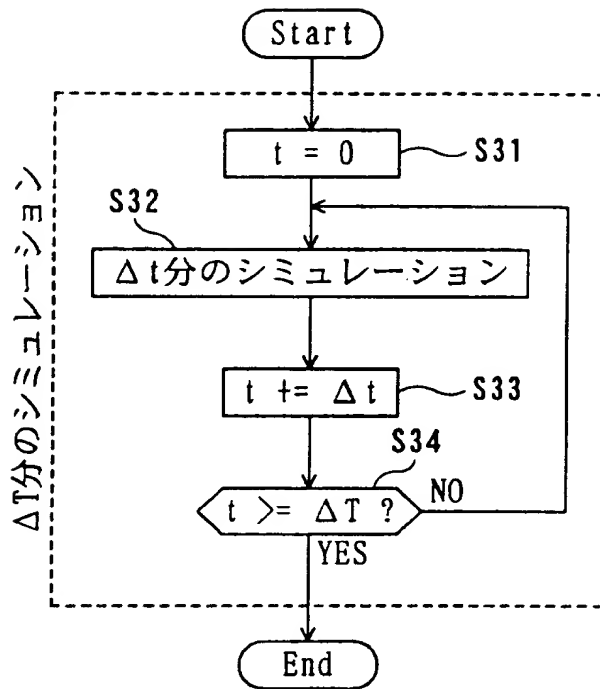
【図 2】



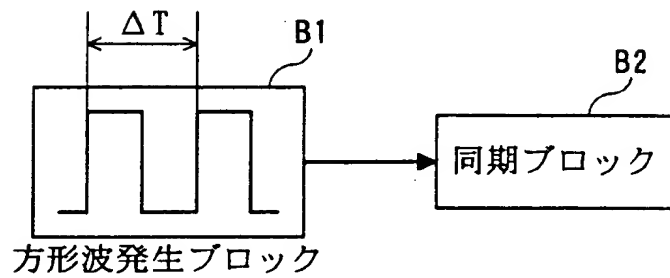
【図 3】



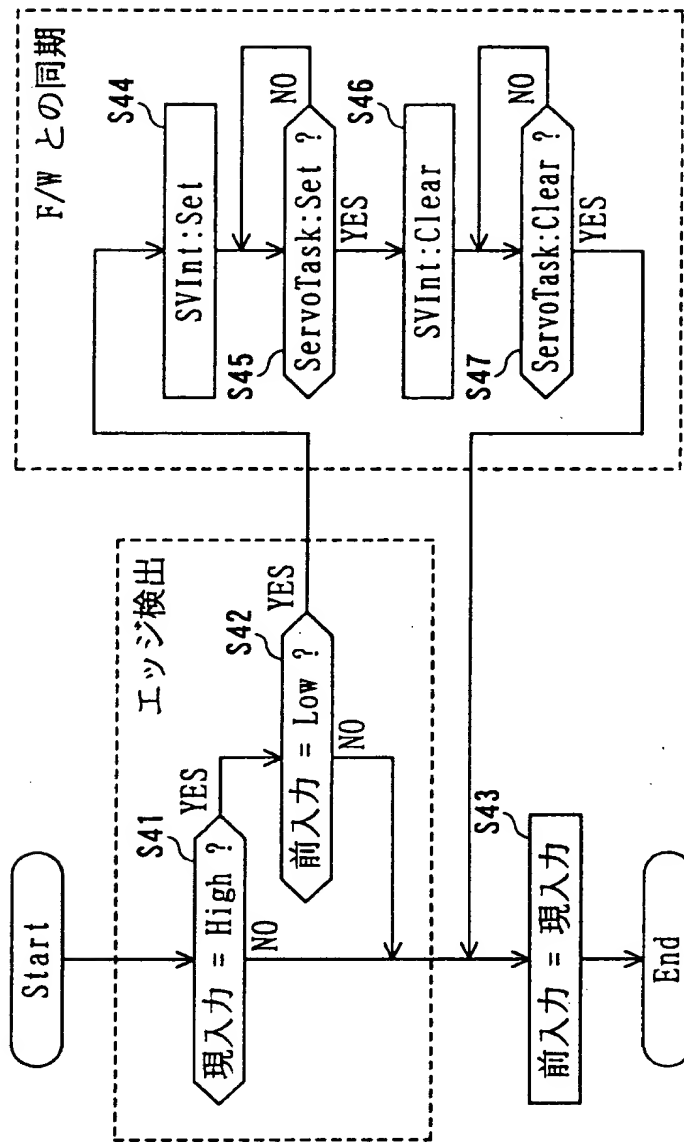
【図 4】



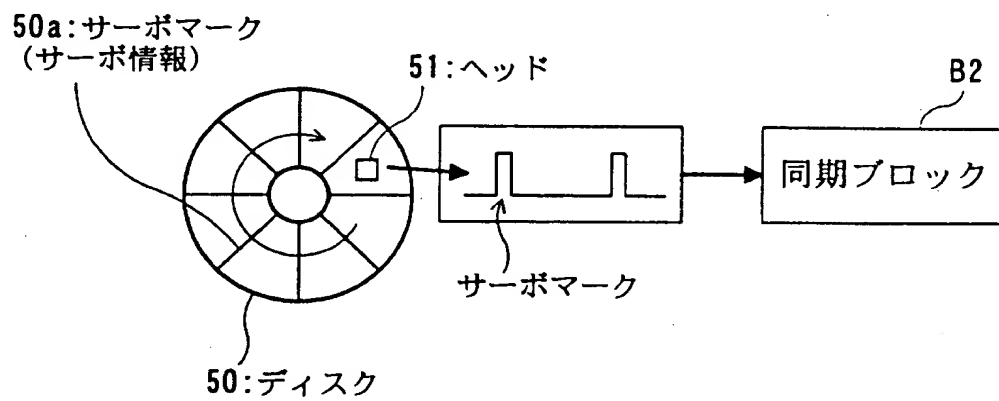
【図 5】



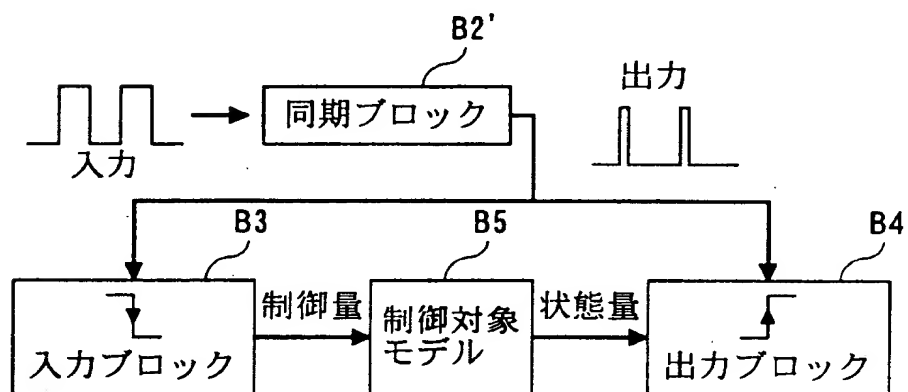
【図 6】



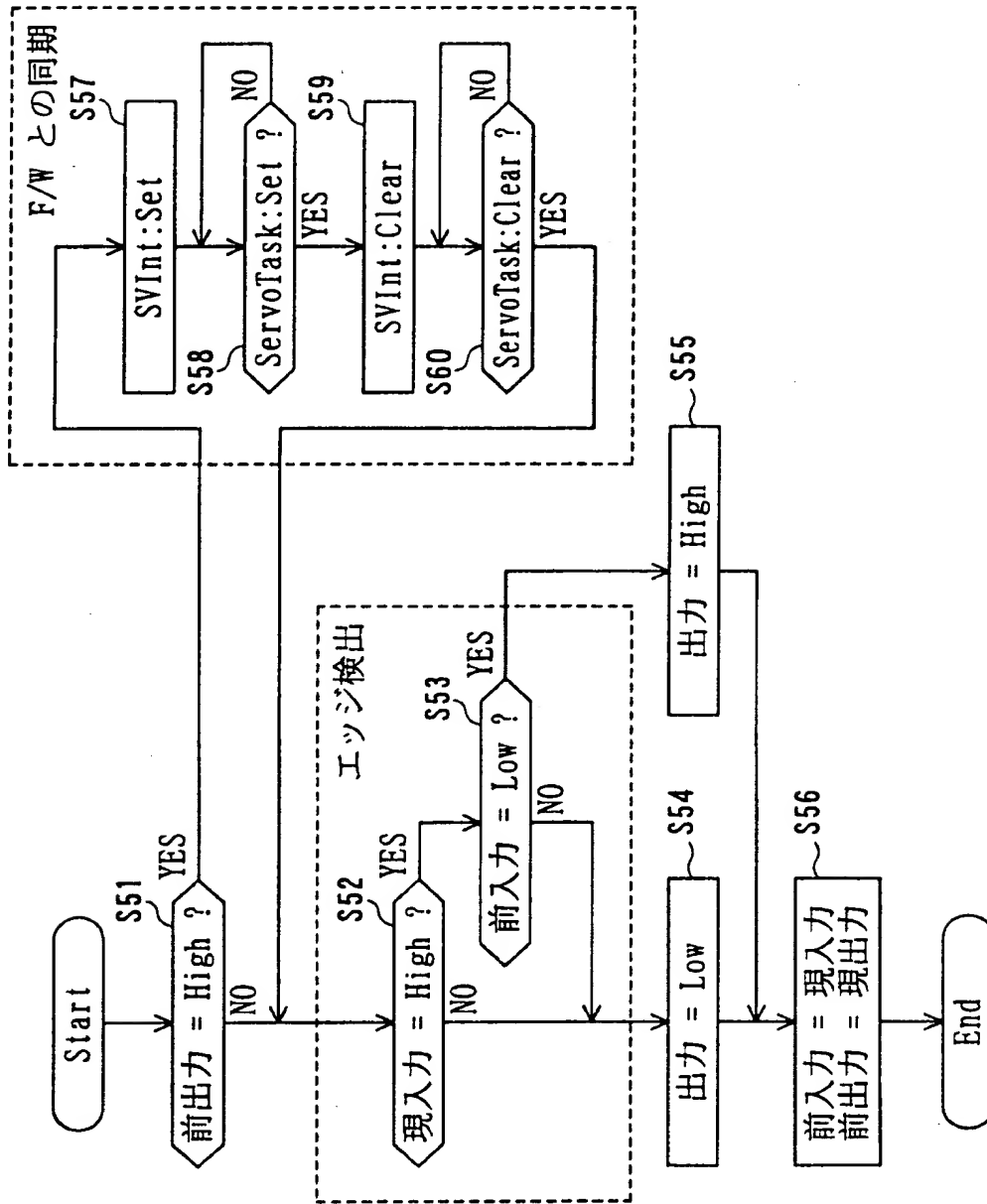
【図 7】



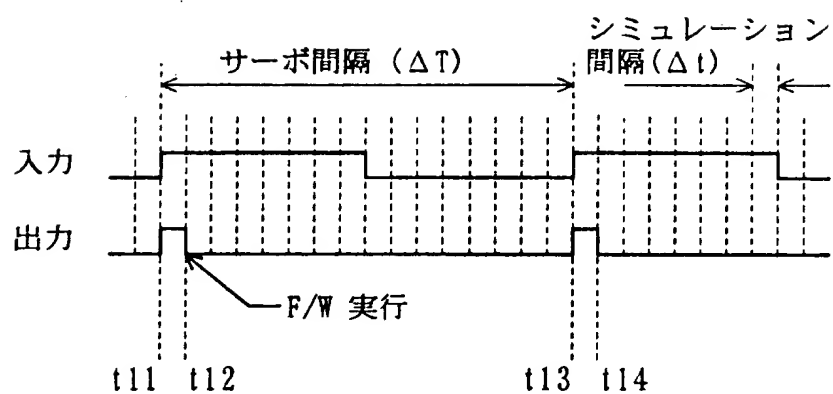
【図 8】



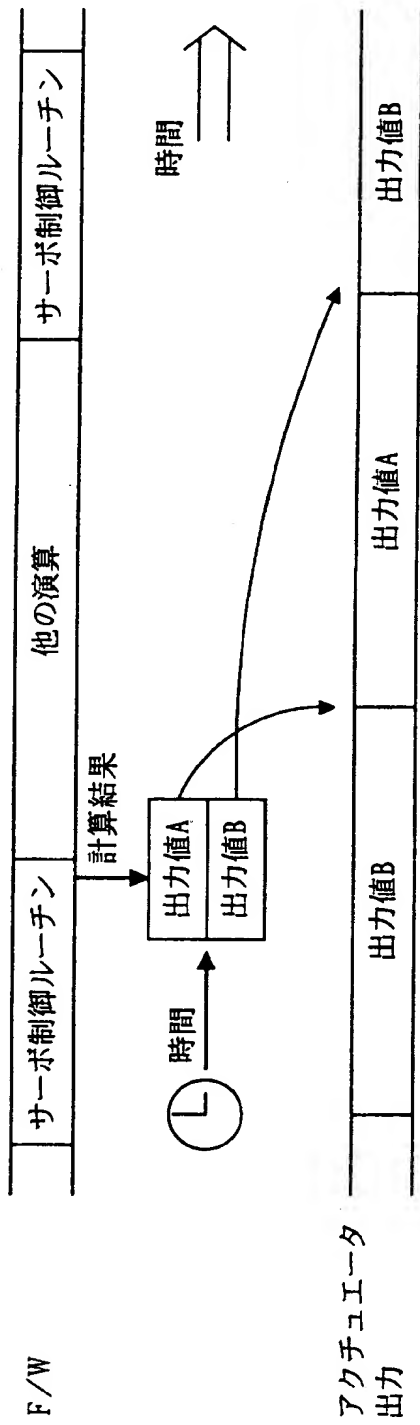
【図 9】



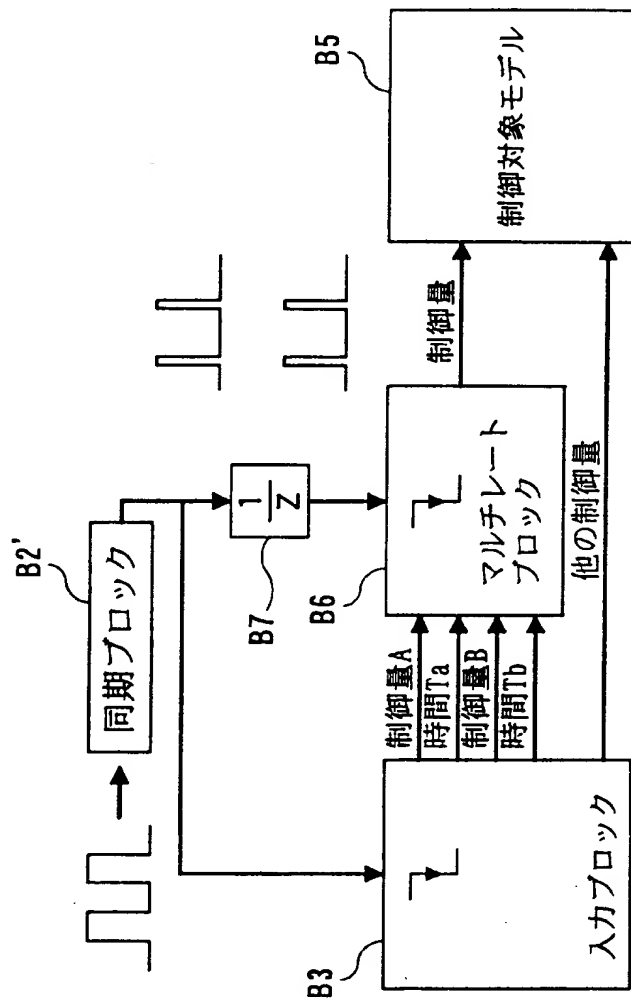
【図 1 0】



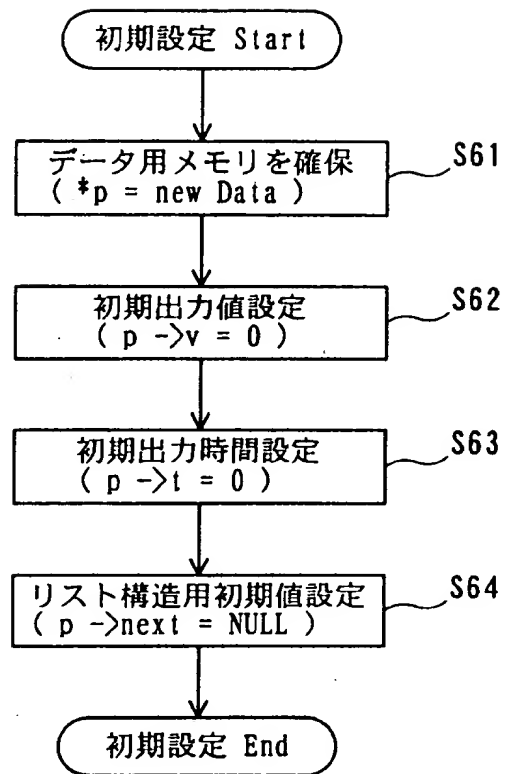
【図 1 1】



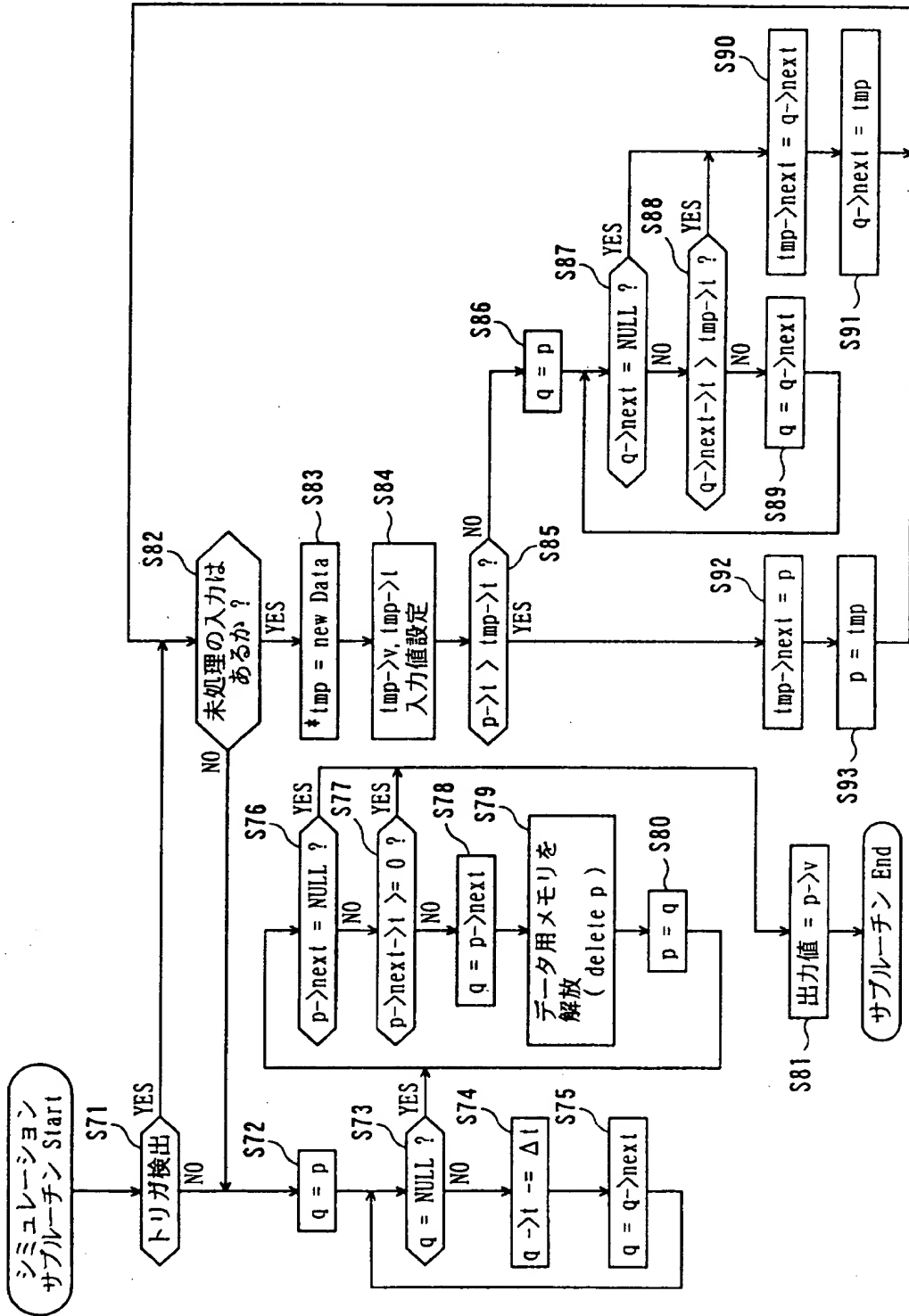
【図 12】



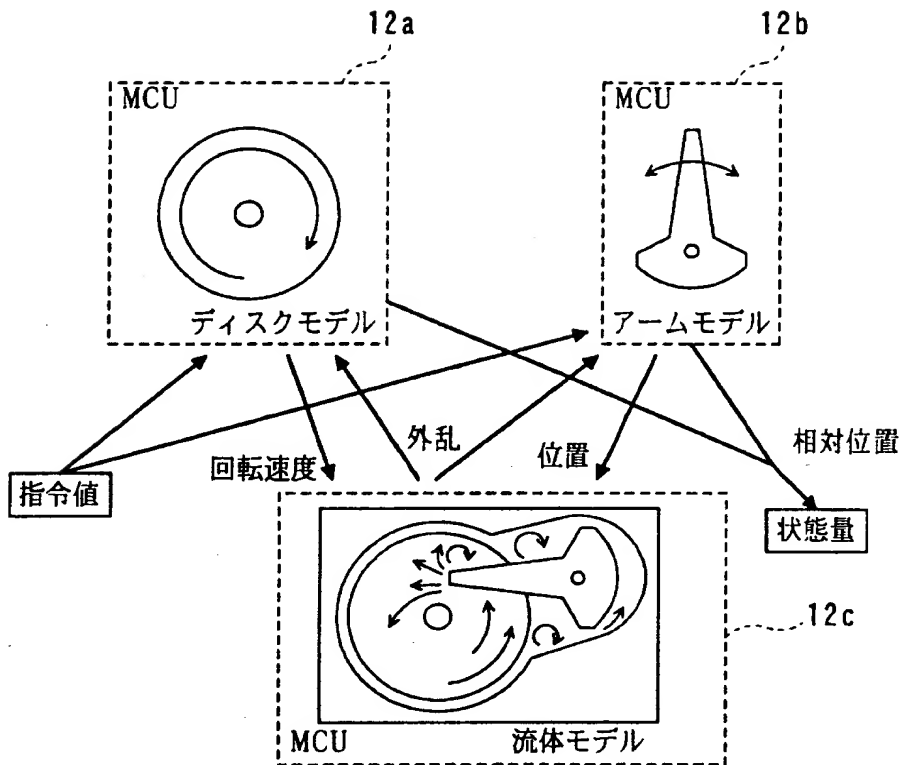
【図 1 3】



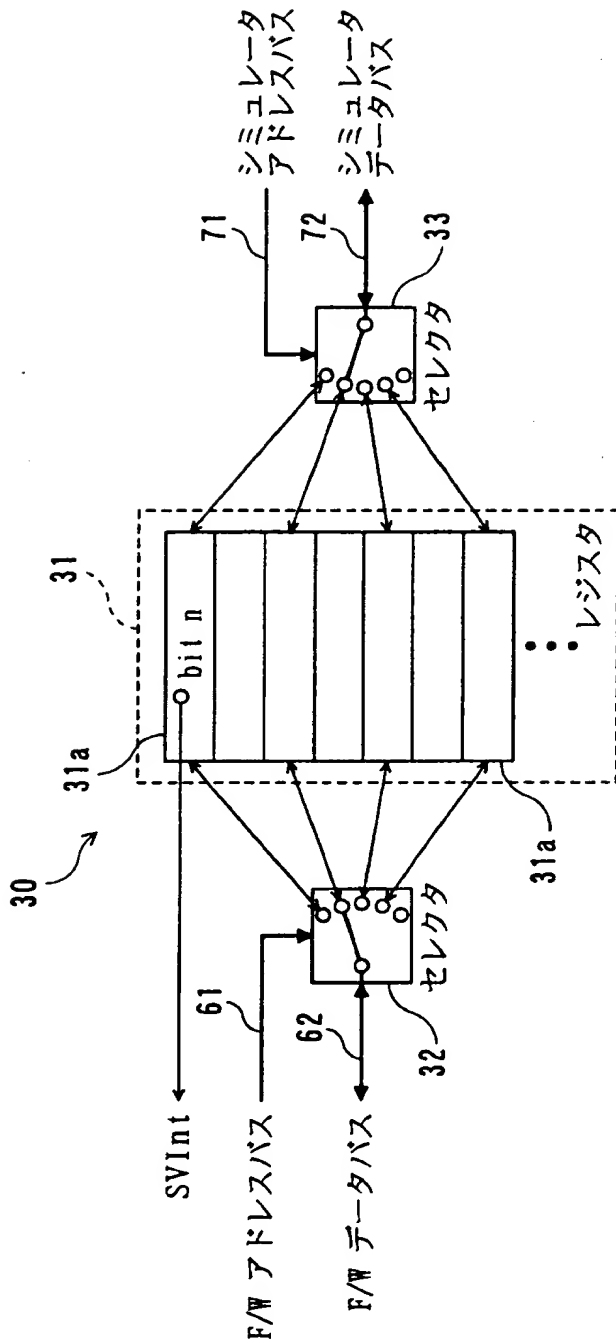
【図 14】



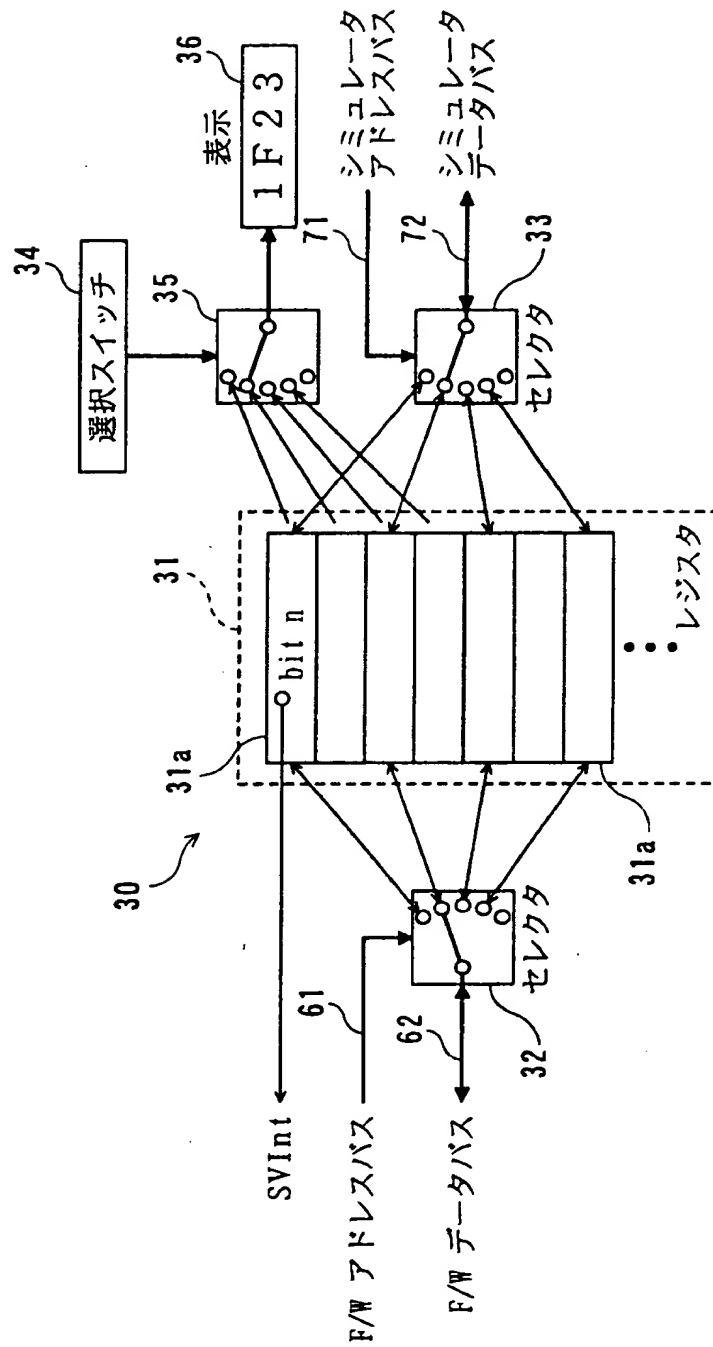
【図15】



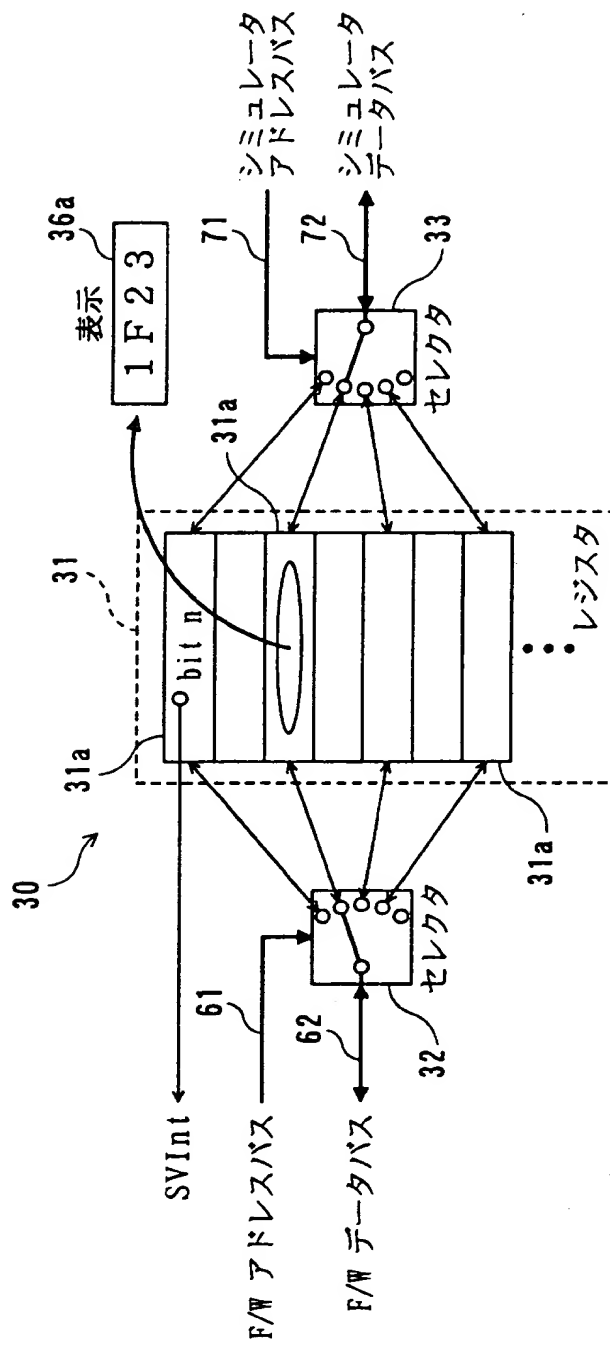
【図 16】



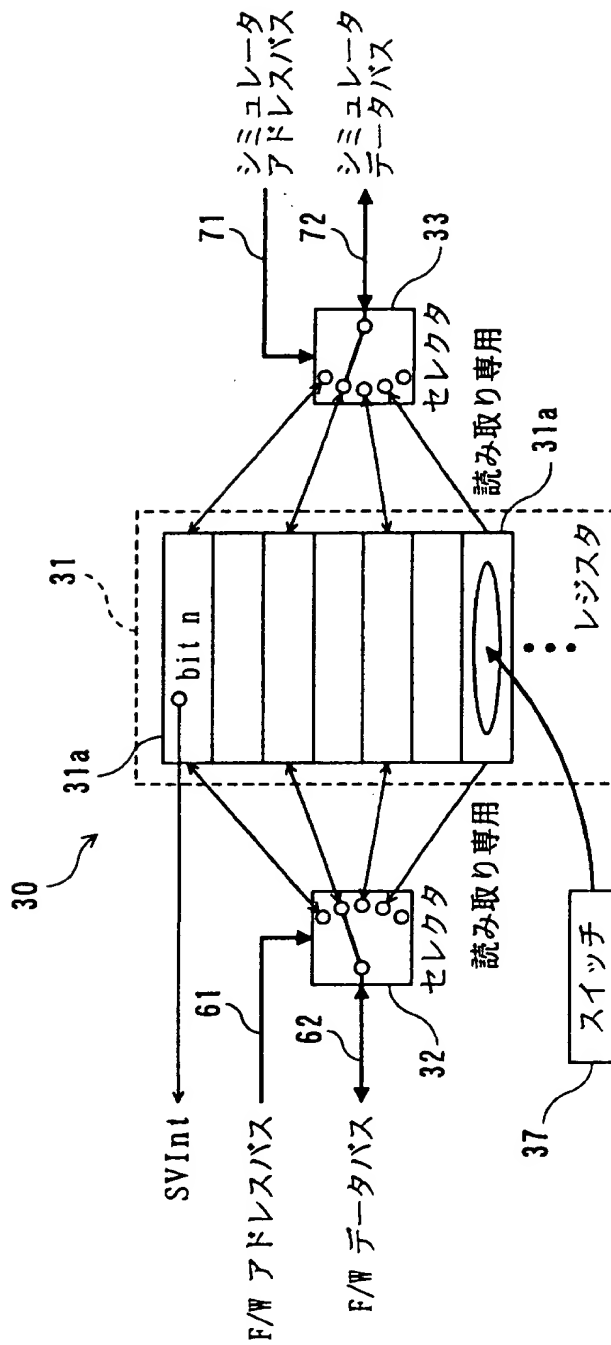
【図 17】



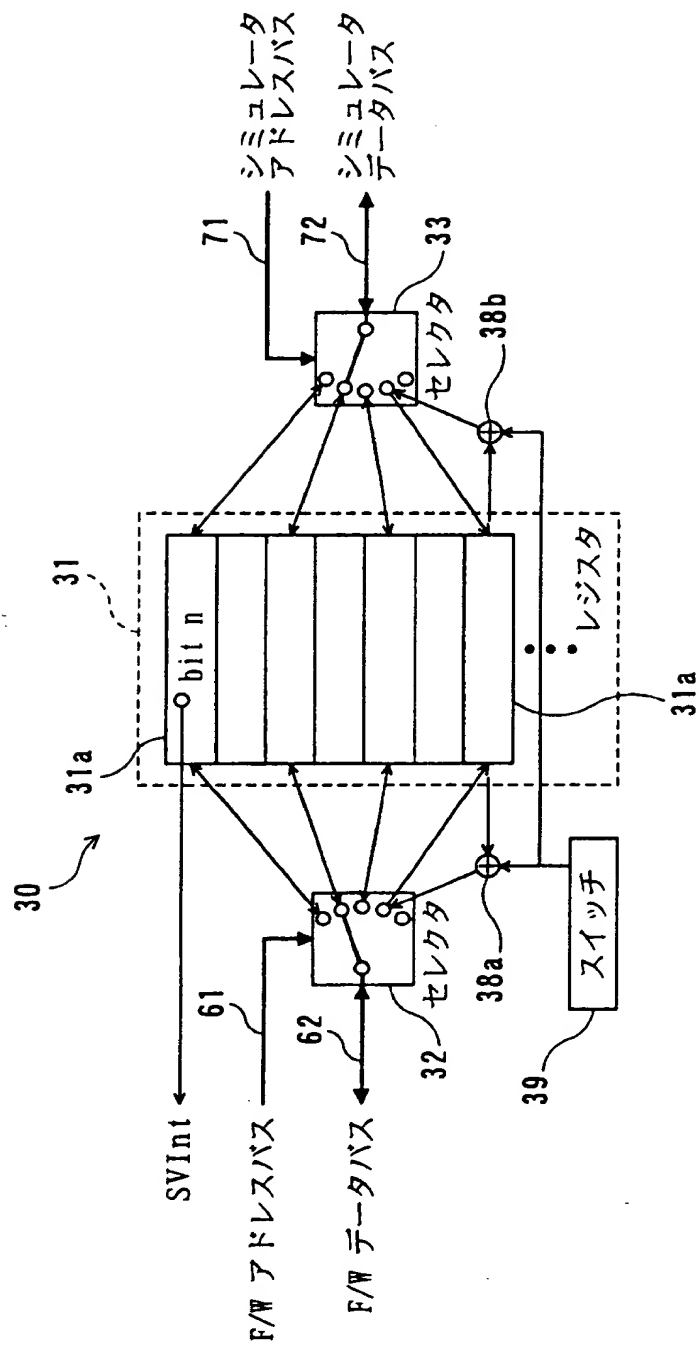
【図 18】



【図 19】



【図 20】



【書類名】 要約書

【要約】

【課題】 実際のメカを用いることなく、比較的小型で応答の速い製品（機構）を制御するための制御プログラムの開発・デバッグを行なえるようにする。

【解決手段】 シミュレーション周期を制御周期よりも短く設定し、シミュレーション部 2 0 が、制御周期に対応する時間に亘ってシミュレーション周期でサーボ機構の動作をシミュレートし、そのシミュレーションによって得られたサーボ機構の状態量を中継回路 3 0 へ出力するとともに、シミュレーション制御部 2 2 が、状態量が中継回路 3 0 に保持されると、シミュレーション部 2 0 を応答待ち状態へ移行させるとともに制御プログラム実行部 1 0 による制御量の算出動作を開始させる一方、制御量が中継回路 3 0 に保持されると、制御プログラム実行部 1 0 を応答待ち状態へ移行させるとともにシミュレーション部 2 0 によるシミュレーション動作を開始させるように構成する。

【選択図】 図 1

出 願 人 履 歴 情 報

識別番号 [0 0 0 0 0 5 2 2 3]

1. 変更年月日 1 9 9 6 年 3 月 2 6 日

[変更理由] 住所変更

住 所 神奈川県川崎市中原区上小田中4丁目1番1号

氏 名 富士通株式会社